A PUBLICATION OF THE INTERNATIONAL FUNCTION POINT USERS GROUP

MetricViews

ifpug.org

March 2020

Vol. 14 • Issue 1

inside

YOUR VIEWS ON METRICS

IN THIS EDITION

Here is a snapshot of the informative articles you will find in this edition of *MetricViews:*

20/20 Hindsight: Tips for Software Measurement Success
(Carol Dekkers)

The author reflects on both our accomplishments and our missteps, and shares with us some nuggets of wisdom to increase our future success. Based on two decades of software measurement experience, Carol Dekkers has compiled a list of 10 tips that will help increase success for software measurement programs.

The Excellence Determining ILF and EIFs
(Antonio Ferre Albero)

To measure with a high level of accuracy is one of the most challenging tasks when no conceptual or logical data model exists or it is not as correct as desired.

***MetricViews* Talks with Talmon Ben-Cnaan**

MetricViews had a recent conversation with Talmon Ben-Cnaan. He is the chairperson of the Non-Functional Sizing Standards IFPUG Committee and is part of the team that created the IFPUG Software Non-Functional Assessment Process (SNAP) method.

Size as a Factor in Test Estimation with a Focus on Test Case Points
(Thomas Cagley)

Thomas Cagley introduces us to an emerging sizing technique called "Test Case Points." Test Case Points are a unit of measurement generated from the testable requirements based on a set of rules. In addition, he shares with us several other size techniques for testers that are in use in many organizations.

Non-functional Complexity Threshold in Function Point Delivery Rates
(Carlos Eduardo Vazquez)

In this article we read about improving productivity risk distribution among business areas and software development organizations with intrinsically high delivery rates due to implementation and design responses to non-functional requirements in organizations with mature IT governance practices.

Measurement Origins, When and Then, Provoking Us to Think Again
(Joe Schofield)

Joe Schofield points out that we have been successfully advancing the development and use of software metrics over time. He includes an interesting review of some of the origins of measures and ponders the question, who and why would someone spend time measuring.

Christine Green

Message from the President

This issue of *MetricViews* is all about the experience and best practices with metrics. As it was stated in the request for articles: “We learn from our experiences, good or bad. We can also learn from other people as they share their knowledge and experiences. IFPUG has a highly valued user community and one of the key elements to its success is the sharing of information.”

IFPUG has for me always been such a great place for learning and sharing knowledge. This has been the greatest source of information about what is current and relevant within the IT industry from a process and performance perspective. What gets defined as a best practice and the lessons learned help us to stay current and relevant.

This edition of *MetricViews* is called Your Views on Metrics. With the quality of the articles and experience of the authors, *MetricViews* will once again be a collection of high experienced best practices as well as introductions to areas or processes that I have never heard about before.

For a long time, we have been talking a lot about Agility—and reasons for including it from a business perspective—Business Agility. In all areas we need to be able to respond quickly to changes, changes in strategy or from attack from the outside. We need to be resilient and have the ability to reconfigure, recover and repurpose our business. We need to increase the ability to welcome changes.

One of the ways for any business to do this will require digitalization of business processes to support a quick change in the way of working. As an example—flexibility of easily changed data, such as Meta Data (static data), has a quick turnaround in additional workflows and even changes in the functionality of the system. Agility and Business Agility require that software has the right functionality, is flexible and setup with non-functional support that enables the ability to morph quickly—if needed.

My experience is that metrics and measurement can support this—both from the development perspective but certainly also from a business perspective. Contracts based on stable unit price—IFPUG Function Points—can be a way to short cut the decision making for large companies that have IT suppliers. Using size measure to evaluate not only the quality and detail level of requirements—but also an upfront input to how many sprints we need in order to complete the required changes. Business Agility does not remove the need for competitiveness and earnings—using Size Measures for competitive benchmark does decrease the cost of a company’s IT budget.

Metrics and measurement work in my mind perfectly with Agility and Business Agility. Not only in a retrospective manner but in many other areas. Metrics and measurement are absolutely supporting assets to succeed with Agility. So, what is required in order to make Metrics and Measurement current and relevant? My experience is that the Metrics and Measurement need to be actionable. They need to be consistent and repeatable. In addition, in a lot of Metrics we need a normalizer that enables the Metrics to be used—not only within a team, but also by the organization and by the management for qualified decision making and to determine early warnings for intervention or action and finally for the ability to measure things like competitiveness, improvement and quality.

Sharing knowledge and the lessons learned in *MetricViews* are not just matters of release and read. All of the work of writing the articles, reviewing, assessing feedback and assembling the articles for *MetricViews* is done by volunteers. Volunteers are the key to the success of IFPUG. Their willingness to use their mornings, evenings, spare time and family time to bring value to IFPUG is the key. So why do they (we) do that? My belief is that it is a combination of both professional and personal growth. Professional in that working as a volunteer will grow your skills not only for Function Point Analysis or Non-Functional Assessment, but for the whole area of the IT Industry. Personally—because IFPUG volunteers and members have great personalities—IFPUG produces lots of laughter as well as learning.

In conclusion, thanks to the all of our great volunteer authors for sharing your knowledge. Thanks to the Communication and Marketing Committee for being consistent in the release of *MetricViews* and finally—to the readers—I hope you will enjoy yet another example of the power of the IFPUG community and volunteers. ■

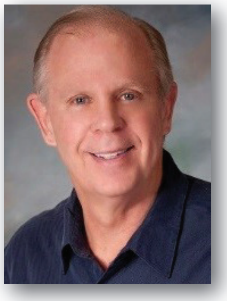
Yours sincerely,
Christine Green
IFPUG President (2019-2021)

MetricViews

Published twice a year by the International Function Point Users Group (IFPUG), headquartered in Princeton Junction, New Jersey, United States

MetricViews March 2020

Editor
David Herron


David Herron

From the Editor's Desk

Welcome to the start of a new decade. It only seems fitting that we begin the New Year on a positive note. We don't often stop to think about how IFPUG manages to continuously provide its user community with a wide array of conferences, trainings, publications, resources and certifications.

All of this is made possible by the dedication of individuals who volunteer their time to govern and support IFPUG. It is an international blend of professionals who find time to contribute to the continued development and advancement of two industry standards; Function Point Analysis (FPA) and Software Non-functional Assessment Process (SNAP).

They believe in and advocate for the IFPUG mission: "The mission of the International Function Point Users Group is to be a recognized leader in promoting and encouraging the effective management of application software development and maintenance activities by providing software sizing standards and other software measurement techniques."

So, let us give thanks to all those who serve IFPUG so generously. THANK YOU to the board members who give of their time to run the business of IFPUG and strive to keep the vision and mission current with the needs of the industry. THANK YOU to the committee chairs who have the thankless job of organizing and managing their committee members. And THANK YOU to all of those individuals who serve on committees. I have been involved with IFPUG for the past 30 years and have served on numerous committees. It never ceases to amaze me the dedication and commitment people contribute to provide services to the IFPUG community. And let's not forget to give a big THANK YOU to all who attend the conferences, the trainings, become certified professionals, make use of the myriad resources and to those who contribute to *MetricViews*. ■

David Herron
Communications and Marketing Committee

20/20 HINDSIGHT: TIPS FOR SOFTWARE MEASUREMENT SUCCESS

By Carol Dekkers

It's customary at the dawn of a new year (and a new decade) to reflect on both our accomplishments and our missteps hoping to glean some nuggets of wisdom to increase our future success.

When it comes to corporations and organizational learning, two famous adages "Hindsight is 20/20" and "History repeats itself" come to mind.

How Does this Relate to Software Measurement?

In the 40 years since Function Points first appeared as a measure of software size, software measurement programs have ebbed and waned, with IFPUG function points gaining in popularity in the 1990s (an Orlando IFPUG conference circa 1994 boasted more than 300 attendees) then fading during recessionary periods. Today, Function Points are again on the rise, and are being included as units of measure in the International Cost Estimating and Analysis Association (ICEAA) Software Cost Estimating Body of Knowledge (sCEBOK) certification. Author and measurement industry guru Capers Jones touts function points as being the "universal software metric."

While the increase in function point acceptance spells good news for IFPUG and function point proponents, software measurement success will still depend on good planning and proper selection of measures, including function points.

Based on two decades of software measurement experience, I've compiled a list of considerations that will help increase success for your software measurement program:

1. Consider the Software Measurement Program as a Product

Measurement is often viewed as a peripheral initiative to software development that starts out with, "Gather up some project data that demonstrates to management what progress is being made in IT." Such an ad hoc approach typically leads to failure as there really is no measurement program per se.

A better approach is to consider software measurement as an Executive Dashboard that can provide answers to specific strategic questions. What data is necessary for presentation? The answer depends on the goals and the questions that management wants to answer.

Goals and questions are the foundation of Dr. Victor Basili's Goal Question Metric (GQM) approach to measurement—whereby one starts with measurement goals first; then identifies the questions that will answer whether or not the goals are being achieved; and then selects the metrics that will provide the answers to said questions.

Using an Agile approach to deliver a software measurement program (including a prioritized product backlog complete with user stories to be delivered incrementally) could be an excellent way to develop the measurement dashboard and make it tangible to management.

2. Start Small and Get the Kinks Out

One of the biggest challenges to measurement program success is making sure that there is a common understanding of measurement and that everyone involved collects the same things in the same way. To a measurement practitioner, the definitions of measurement and data collection may seem straightforward and clear, but to those new to software measurement they are anything but. One of the best ways towards measurement program success is to start small and prove that measurement processes deliver results before rolling out the program on a grand scale.

"Software measurement success will still depend on good planning and proper selection of measures, including function points."

Two relevant historical cases overlooked the importance of starting small and ended up with failed measurement programs:

a) A large telecom company delivered presentations at consecutive IFPUG annual conferences detailing their "Corporate Wide S/W Measurement Program." Each year for five years, a new director presented what he/she called their new corporate wide program—revamped and tweaked so that it would deliver results. In year six, management scrapped the program professing that "Function Points didn't work," when in actuality, the program never took off because it was too big and didn't align with the corporate goals. A small pilot program in year one could have proven (or not) the merits of the program and could have prevented the larger failure.

b) A client of mine hired a group of CFPS consultants (including me) to FP count their projects so that they could fill in the FP values on their corporate scorecard. When presented with requirement documents for a four-phase project, I asked the client which phase(s) of the project he'd like me to count. Incredulously, he replied, "I have no idea, I just know we have to put FP numbers into blank fields on the scorecard so just give me some numbers." Function points were being collected without any checks on their reliability and devoid of contextual project factors. After months of collecting such (incoherent) data, the metrics program was scrapped.

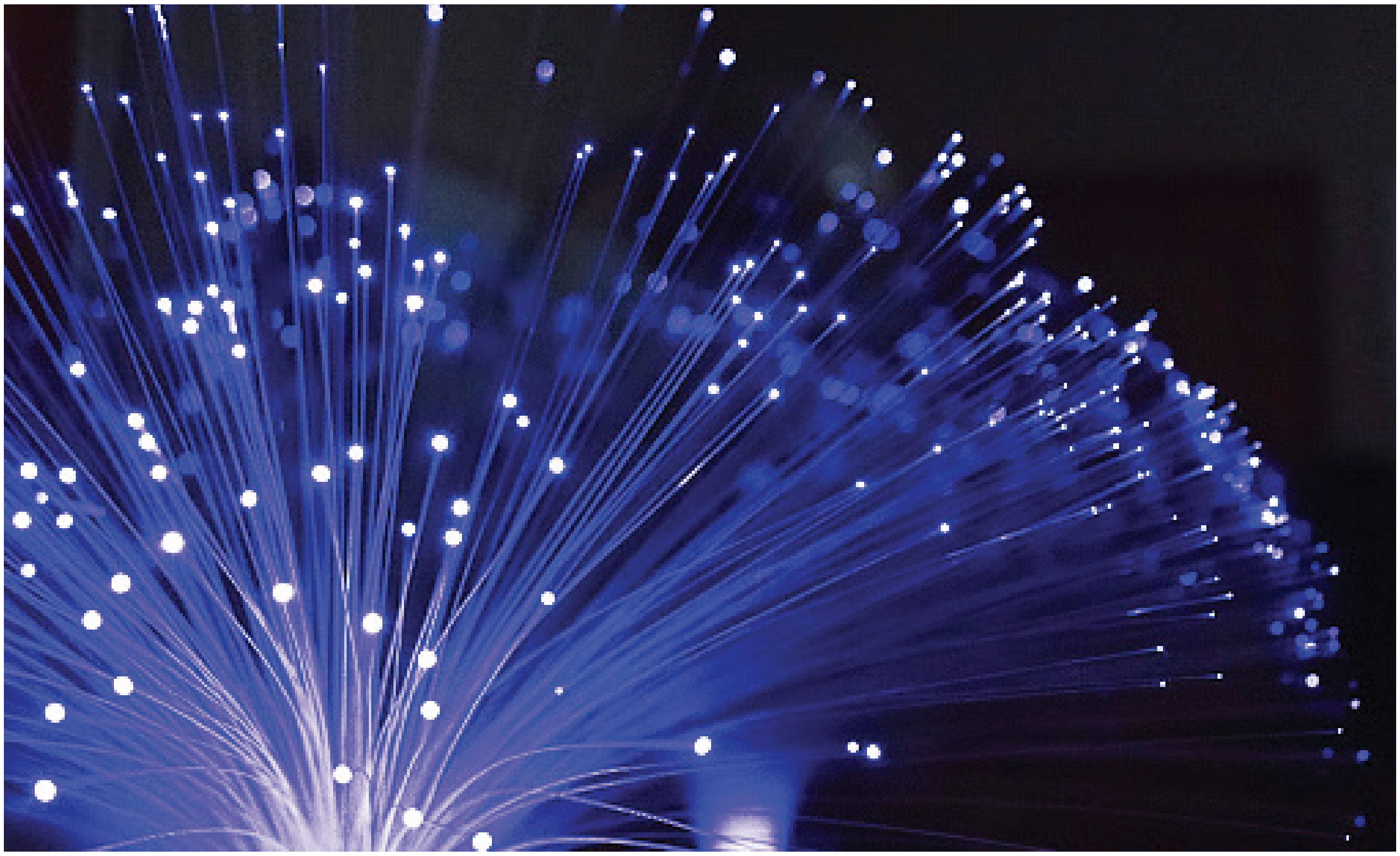
Starting measurement as a pilot project allows you to ensure that you "get the kinks out" early and prove early success (or failure). Much better to revamp a small program than to cancel a large one.

3. Ensure Measures are Meaningful

When presenting metrics, consider the audience who will be seeing them. Dr. Howard Rubin presented this concept as "Audience Analysis" meaning that the measures (and metrics) should have meaning (and therefore value) to the audience who receives them. Measures are not "one size fits all" and should be tailored to meet the needs of the audience. For example, if management needs to see return on investment figures, it makes no sense to present them with a project burndown chart.

4. Plan Metrics with the End in Mind: Show Causality

When No.1 is not an option, management will often ask for "quick and dirty" metrics (meaning: quickly gather up some project data, assemble it into graphs and report some findings



"Many a measurement program has failed because it was too complex."

so we can make decisions)—you may be lulled into compliance. Sure, it's easy to pull together data by gathering project size (Function Points), effort (project hours) and other project characteristics (programming language, type of projects, etc.)—and calculate "productivity" or delivery rate metrics (hours/FP)—but without context or causality, the data won't be useful. Never present a graph or chart that implies causality when the data doesn't support it.

Bar graphs depicting productivity levels from a diverse set of projects are downright misleading because they imply causality that simply isn't there. The very word "productivity" implies that people (teams) are part of the cause of productivity levels, when cause(s) lie in differences in project characteristics (new versus enhancement, package versus custom development, 3GL versus 4GL tools, type of industry, etc.).

5. Consistency Trumps Perfection

Don't wait to gather measures until you have every last detail worked out (that is similar to waterfall software development)—start small, ensure that measures are being gathered consistently and reliably, but don't insist on perfection. Being consistent with the interpretation of function point rules (and applying the measure consistently) is more important than waiting until everyone agrees with the exact interpretation of the rules according to the IFPUG counting practices manual. Arguing over 3 FP on a 5000 FP count is time that could be better spent documenting a count, yet in the quest for perfection, FP counters will often argue. Don't succumb to this illusion of exactness—be consistent and move forward.

And, if you insist that everyone on your FP measurement team is a Certified Function Point Specialist (CFPS) before you begin counting, you'll end up losing valuable data. Consider hiring a CFPS consultant to assist you in getting started and/or engage qualified training (or hire consultants to do your counting if your budget allows) and build up the consistency. While the CFPS designation certifies that the FP counts are done according to the IFPUG counting rules, the exam can be tough for even seasoned (and well-qualified) counters to pass (90% average is needed to pass)—and even excellent FP counters might need to retake the exam before they pass.

6. Allow Critics to Voice Their Opinion

While you are designing and implementing your pilot measurement program, there are sure to be spoken (and unspoken) critics who disagree with your approach or who believe that measurement will unfairly disparage their work. Allow critics to voice their opinions freely and be patient. Some of the most vocal opponents of measurement can become staunch supporters (and could even suggest better ways of doing measurement) when they understand the goals behind the program.

7. Train, Explain, Retrain: The Power of 3

Software measurement to non-measurement practitioners is non-trivial and often non-intuitive—especially when measures such as Function Points are involved. Just as with any other new concept, it often takes a minimum of three exposures before the main understanding takes hold.

Plan to train everyone who will be using, collecting or receiving the measurement data—and do it in frequent, small, bite size, repeatable pieces. While Function Points represent the Functional Size of software, this is counter-intuitive to programmers or software developers and it takes a paradigm shift to realize that they do not represent the entirety of software size. Check back with students after training is completed to ensure that it delivered adequate training transference to equip people to do what you expected with measurement.

8. KISS: Keep it Simple and Supporting

When challenged to design a successful measurement program, many practitioners (and FP counters) go all out and design a comprehensive program that can actually get in the way of rather than support software development. Measurement should always support the software development effort by gathering data "alongside" of it rather than impeding its progress. Many a measurement program has failed because it was too complex and encumbered by collection processes that interfered with (or took away) valuable time that could have been spent on software development.

If your software measurement program becomes the end instead of the support for software development (i.e., gathering data to support process changes), the program will most likely fail. Software measurement must deliver value on its own and not detract from the main business of delivering better software.

9. Use ISBSG or Other Data to Augment Your Own (Lack of) Historical Data

If your goal for software measurement is to improve your software estimating, it may take years to develop enough historical project data on which to do better estimates. In the meantime, you may want to rely on established industrial databases such as the International Software Benchmarking Standards Group (ISBSG) Development and Enhancement database of more than 9,000 completed projects, or use a software estimating tool that contains historical project data similar to your own (such as SEER SEM, QSM's SLIM or Price S). In this way, you can validate your own collected data at the same time as you do new estimates and compare the results as you build your own historical database.

10. Don't Go it Alone

While it may seem daunting (or for some, a unique and exciting challenge) to design a software measurement program that will succeed and be relevant to your company or organization, don't be tempted to recreate the wheel (so to speak) on your own. There are numerous global organizations that you can join for a low cost with a mission to support software measurement and that can connect you with a network of like-minded professionals. The overall software measurement community is still growing and by and large is very accepting of new people with new ideas.

Of course, I recommend IFPUG (www.ifpug.org), and you may also want to consider:

- International Cost Estimating and Analysis Association (ICEAA) at ICEAAOnline.com
- ISBSG (ISBSG.org) is open to corporations and measurement associations
- and there are a number of other international software metrics groups (often country and language specific).

Over the coming years, more measurement programs will be implemented and more money will be invested (hopefully) to reap the rewards from data analytics. Success will come from proper planning and appropriate data analysis. I wish you success in your measurement endeavors. Please share your successes (and failures) through this and other communities. Together, a rising tide floats more boats. ■

About the Author:

***Carol Dekkers**, PMP, CFPS (Fellow), P.Eng., CSM, is founder of Quality Plus Technologies Inc. and consults with companies that want to succeed with software measurement and functional size measurement. Carol conducts Function Point training worldwide for clients and conferences and is one of the leading FP authorities and ISO standards measurement experts. Ms. Dekkers is a past president of IFPUG, a former director of Communications and Marketing and currently serves as the chair of the Industry Standards Committee. (Email her at dekkers@qualityplustech.com.)*

The Excellence Determining ILF and EIFs

By Antonio Ferre Albero

To measure a project with a high level of accuracy and excellence, adaptation or an enhancement is a must for a CFPS or CFPP.

Identifying correctly the Logical Files (ILF and EIF) is perhaps one of the most challenging tasks when no conceptual or logical data model exists or it is not as correct as desired.

In the opposite way, Input or Output transactional functions (EI, EO and EQ) by general rule have less discussion because in almost all the cases the evidences are clearly visible; e.g., an input screen, a process that imports a file, a report, a process that generates a file... Perhaps the main discussion can arise, again, determining what is considered a file (FTR; File Type Referenced) for those input and output transactions, and to refine the "process" concept. In addition, some small debate about the concrete number of DETs can exist, but those DET debates, in most of the cases, can be skipped because the size will be exactly the same if the ILF/EIF has 71 or 72 DETs. The exception is when the DET number determines a change in the process complexity from low to average, from average to high or vice versa.

"It is important to emphasize the words "user view" because sometimes a different understanding can be found applying the "user view" concept."

It is well known, even for the people with relatively little experience counting Function Points, that a logical file is something different to a physical file. Sometimes technical or functional experts are more familiar with objects that can be touched, such as a physical file or a database table, for example; "logical" information is a little bit more abstract.

Obviously, those differences are essential because the IFPUG Function Point Counting Practices Manual (CPM) is based on logical files.

Even it is not new that Function Point analysis uses concept of a logical data entity, from the user point of view, that can be grouped in one or more logical files. Moreover, it is important to emphasize the words "user view" because sometimes a different understanding can be found applying the "user view" concept.

It is not unusual to see that a logical data model does not exist, in small as in non-small companies or projects. This situation could be improved but it is the real-life situation in many cases. In other cases, instead of this "user view entities" are reflected directly in the database tables or files.

Not having the correct information from the user point of view, or not reflecting it in the proper project documentation, can result in non-accurate counts. To consider a new superfluous/extra average complexity Internal Logical File (instead to consider that this is not really a new file because it might be comprised into an already existing file) will be counted as 10 FPs. This 10 FPs size will be exactly the same as creating two new average External Outputs (2 x 5 FPs).

Determining that an application has five outputs instead of three, or three instead of five, is very easy and the different parties involved (a CFPS, a user, IT technical people from the customer side, IT provider, etc.) will most likely arrive at the same conclusion after gathering all the information and after applying the CFPS to the correct criteria (criteria=method, output concept, boundary cross concept, process concept, etc.), but talking about the "files" concept, to arrive to the same point may not be so easy.

Why? Sometimes, different people with different disciplines (not technical, technical, CFPS experts) may see the situation differently. This may be the result of the information not being completely written, or the written information reflects more concrete technical aspects than functional (user eyes). Some parties could be interested in having higher or lower counts and even this last argument can increase artificially or decrease the number of files.

Special attention may be put into applying the part 3, chapter 2, of the CPM, step 1 "Identify Logical Files" to determine correctly the files from the user point of view, and step 4 "Identify Record Element Types" for identifying if different entities are one logical file or more than one.

Those two steps are essential and, in spite that it seems obvious, it is crucial to remember [with special attention to the first and second point], that:

- A logical file is a logical group of data as seen by the user.
- More than one data entities can derive into one logical file.
- Information contained might be recognizable or required for the user.
- Entities not maintained by any application are excluded.
- Code data entities are excluded.
- Entities without attributes required by the user are excluded.
- Entities created just for technical reasons are excluded.

To have all of the above points totally clear, to ensure that the answers are based on the word "user point of view" and without any kind of technical influence (synonym in some cases to a pseudo contamination) is essential for arriving to the correct and accurate results. ■

About the Author:

Antonio Ferre Albero *(Valencia, Spain) has more than 30 years of experience in information technology (IT), project management and metrics for private companies, government and large IT companies. He is CFPS accredited, has been member of different IFPUG committees for years and is currently the IFPUG CMC chair. Antonio is project manager at GFT, a European company with offices in 15 countries focused on innovative IT solutions. He specializes in a variety of disciplines including project management, quality and CMMI, metrics, functional size and Function Points, productivity, benchmarking, estimation processes, technology strategies, Db2 and Oracle, databases and big systems. As a senior technologist and project management passionate, he applies best practices to insure IT helps organizations and their employees. Hundreds of Antonio's technical articles have been published many times in newspapers and other print publications.*

MetricViews Talks with Talmon Ben-Cnaan:

"I AM PROUD TO BE PART OF IFPUG AND LOVE TO CONTRIBUTE TO IT. I LOVE WHAT I DO."

MetricViews had a recent conversation with Talmon Ben-Cnaan from Israel. Talmon has been the chairperson of the Non-Functional Sizing Standards IFPUG Committee since 2012 and is part of the team that created the IFPUG Software Non-Functional Assessment Process (SNAP) method, recently recognized as worldwide standard IEEE 2430. He has promoted SNAP along a path from a need within the measurement community to an international standard by leading the IEEE project of software non-functional sizing. Talmon is a Quality and Testing measurements manager at Amdocs. He led Amdocs Measurements Department and was responsible for implementing Function Points in his company. He is an international speaker, addressing the importance of the SNAP method and IT sizing in different parts of the world such as India, Brazil, Italy and Spain.

Talmon, what was your first contact with IFPUG?

It all started when I led the measurements and benchmarking team in Amdocs. When the company decided to use Function Points Analysis, I was assigned to implement the method in the company and provide benchmarking. We started counting FP with a consulting company, and, in parallel, built a team of experts. I joined IFPUG to gain the knowledge (and the relations) and to ensure that we are doing the right thing.

Could you please tell us what the IFPUG SNAP standard method is?

IEEE 2430 is an international recognition of SNAP (Software Non-functional Assessment Process) as a software sizing method to size the non-functional aspects of a software project. The standard describes how to size the non-functional requirements and explains how to use the measurement for more accurate estimation, and for calculating project performance and benchmarks.

You are one of the fathers of the SNAP method. Could you please speak a little bit about the SNAP history?

I came to know about the need to have non-functional size by chance, and I was curious. The IFPUG committee started by collecting cases in which FPA does not cover the effort, and studying the definitions of non-functional requirements. At first, we could not align the two lists into one. The solution came by separating the non-functional requirements from the SNAP sub-categories. The sub-categories define how Non-Functional Requirements (NFR) are met, therefore they can be used for sizing.

A draft of the manual was reviewed by 80 volunteers. After refining the manual, in 2012, IFPUG conducted a Beta test. SNAP Data was collected from 48 projects from Brazil, China, France, India, Italy, Mexico, Poland, Spain, UK, and the U.S., covering the aerospace, automotive, banking, government, fast moving consumer goods, financial services, insurance, manufacturing, systems integrators and consulting, tele-communication and utility industries. The Beta test showed high correlation between SNAP size and effort, and the SNAP manual was published worldwide. Later, we added rules and guidelines on how to size the functional and non-functional requirements without overlap or a gap between the two sizes.

What are the benefits that SNAP brings to the IT community?

To accurately estimate the effort needed for a software project, one needs to measure three aspects of the project: the functional size, the non-functional size and project tasks that cannot be expressed by size. Without SNAP, the non-functional requirements are not measured.

Recently SNAP has become an IEEE standard. What are the next steps?

Like any new method, we need to expand the use of SNAP and to ensure that experts are confident with the benefit SNAP brings. We intend to reach people who can benefit from having the two size methods, such as project managers who control the budget, procurement managers and contract managers.

How do you feel about how companies apply the Functional Size and Non-Functional size concepts?

I am confident that companies will benefit from applying SNAP, and I wish they share with our committee both their issues and their success stories. I am sure that we can further improve SNAP, based on users' feedback.

You have recently been nominated IFPUG volunteer of the year. What does it mean to you, being a volunteer in one of the most prestigious organizations regarding IT metrics and IT sizing methods in the world?

I am proud to be part of IFPUG and love to contribute to it. I love what I do. In the past, I was a mechanical engineer. Mechanical engineering has existed more than 300 years and is very mature. The software industry is a young industry, which lacks some solid, well-established processes. The software measurement process is still to be developed.

Your dream regarding IT metrics?

That measurements are common and used by all. That software measurements are perceived as a part of every project—not as a burden but as a natural part of the project's assets. ■

TI Métricas

Find where you are. Go where you want to.

www.metricas.com.br

SIZE AS A FACTOR IN TEST ESTIMATION WITH A FOCUS ON TEST CASE POINTS

By Thomas Cagley

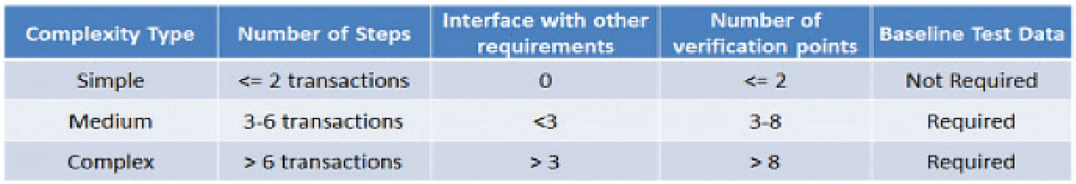
Independent testing groups are often asked how long and how much effort is required to test a piece of work. Several size estimation techniques are actively in use in many organizations. Each of these techniques begins by deriving size either based on a set of rules or through relative sizing. Size, once derived, is used to estimate effort. Effort is then used to generate cost, staffing and duration estimates. An emergent sizing technique is “Test Case Points.”

Test Case Points are a unit of measurement generated from the testable requirements based on a set of rules. The process is straightforward:

1. Identify the testable requirements in a piece of work. Use Cases or technical requirements documents are used for identifying testable requirements.
2. Identify the complexity of each testable requirement. Test case points evaluate four factors to determine complexity:
   a. The number of test steps. The number of execution steps needed to arrive at an expected (or unexpected) outcome after all preconditions have been satisfied.
   b. The number of interfaces to the other requirements.

A simple count of the number of interfaces in the test case.

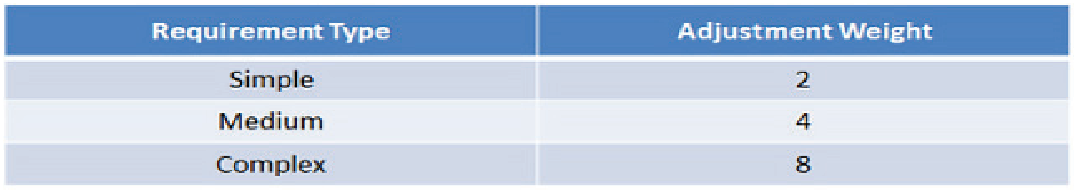
c. The number of verification points. A simple count of the points in the test case that the results are evaluated for correctness.



Complexity Type	Number of Steps	Interface with other requirements	Number of verification points	Baseline Test Data
Simple	<= 2 transactions	0	<= 2	Not Required
Medium	3-6 transactions	<3	3-8	Required
Complex	> 6 transactions	> 3	> 8	Required

d. Need for baseline test data. An evaluation of whether data needs to be created to execute the test case.

Once all of the simple, medium and complex test cases are



Requirement Type	Adjustment Weight
Simple	2
Medium	4
Complex	8

identified, they are summed by category.

3. Weight each category.
4. Sum the weighted categories together to yield the total test case points.

The goal of test case points is to use size to generate an estimate. Every version of test case points I have worked with uses a set of factors to adjust the size as part of the sizing process.

1. Develop an estimation adjustment weighting based on a set of factors (for those familiar with IFPUG Function Points this adjustment is a similar process to the one for determining the value adjustment factor). The factors are:

1. Count or Single Factor Adjustment Factors

Factor 14 – Operating System Combinations (simple count)

Factor 15 – Browser Combinations (simple count)

Factor 16 – Productivity Improvement from Second Iteration Onwards (percentage)

“The goal of test case points is to use size to generate an estimate.”

2. Factors that leverage a combination of fixed factor and complexity weighting

Factor 1 – Domain Knowledge & Complexity

Factor 2 – Technical Know How

Factor 3 – Integration with other Hardware Devices such as Handheld Devices, Scanners, Printers

Factor 4 – Multi-lingual Support

Factor 5 – Software/Hardware Setup

Factor 6 – Environment Setup

Factor 7 – Build Management

Factor 8 – Configuration Management

Factor 9 – Preparation of Test Bed

Factor 10 – Stable Requirements

Factor 11 – Offshore/Onsite Coordination

Factor 12 – Test Data Preparation

Factor 13 – Network Latency

5. Generate an estimate using the following formula:

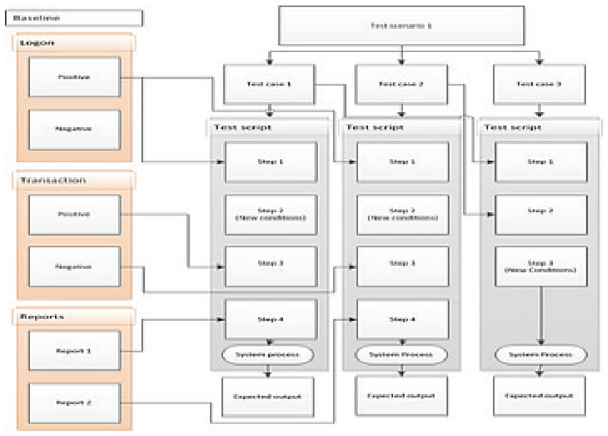
Weighted Test Case Points X Adjustment Factor X Historical Productivity Rate

In many cases, organizations generate estimates for types of work separately using the adjustment factors that would affect the type of work. An example of a type of work is test case generation. Factor 5, software/hardware setup, would not be predictive of the effort for setting up test cases.

The process for deriving test case points is fairly straightforward (steps 1 – 4). The process of turning the test case points into an estimate is more complicated. In order to understand test case points, we will develop a short example and examine the strengths and weaknesses of the process —some which are very apparent and others that are not.

An Example: Strengths and Weaknesses

Jeremy Berriault provided an example from this presentation at QAI Quest 2017 for us to count test case points. Jeremy, QA Corner, indicated baseline data was required to effectively run the three test cases in his example



The logon, transaction, reports and expected output blocks represent verification points. The arrows from one test case to another represent interfaces steps in the test. The results of the count are as follows:

Test Case	Number of Steps	Interfaces	Verification Points	Baseline Test Data	Complexity
test case 1	4	0	4	required	medium
test case 2	4	0	4	required	medium
test case 3	3	2	3	required	medium

Deriving the complexity leverages the following chart:

Requirement Type	Adjustment Weight
Simple	2
Medium	4
Complex	8

The test cases generate three medium test cases or 12 unadjusted use case points (3 x 4 = 12). This represents the output of steps 1 to 4 of the basic test case point counting process. Step 5 requires the evaluation of the 16 adjustment factors. For example, a set of test cases that required multi-language support would require more effort for a set of test cases that requires only one language support.

Strengths:

1. Rules Based: Rules increase consistency and enable cross-team or project comparison.
2. Identifies Complexity: Test cases that are complex based on the rules are targets for simplification. Complex test cases are harder to execute (whether automated or manually) and evaluate.
3. Generates Thought and Conversation: The process of sizing forces practitioners to think about and talk about what they are testing.
4. Testing Specific: Test case points relate only to the activities performed by testers.

Weakness:

1. Not Available Early: Counting test case points requires a level of detail that is often not available early in Agile efforts (IFPUG works better).
2. Testing Specific: Test case points can only be used to predict testing activities, in cross functional teams just sizing one part of the lifecycle is not as useful.
3. May Not Be Predictive: There is no published industry data that proves test case points are predictive of the testing effort. Many test case point estimation approaches are driven by weak correlations.
4. Only Testable Requirements: Not all requirements are "testable;" however, non-testable or non-functional requirements still require effort to evaluate.
5. The Relative Adjustment Factors: Some of the adjustments require interpretation rather than explicit criteria-based evaluation. Subjectivity will likely creep into the process.

Test case points are a useful sizing technique. Test case point users need to answer two questions: whether the value of the information generated outweighs the effort needed to generate the number, and when the information to count test case points is useful.

"Test case points is only one approach to determining the size of work that needs to be tested."

Other Sizing Techniques for Testers

Test case points is only one approach to determining the size of work that needs to be tested. The other measures fall into three broad categories. The categories are:

Physical Measures. This category represents count tangible "things" like requirements or test cases. The assumption is that there is a relationship between the count of the physical item and the effort or the duration of testing. For example, there might be a relationship between the number of test cases

needed for a project and the amount of effort needed to execute and review those test cases. Measurement approaches that count physical items are generally an easy approach to generating a measure or metric. The most immediate problem with counting physical things is that individual items are generally not the same size. Therefore, simple counting does not reflect the range of sizes.

Functional Measures. This category uses a set of rules to determine software size by assessing the software based on a set of rules focusing on delivered functionality. IFPUG Function Points (and other function point methods) are examples of size measures in this category. The assumption made when using this category of metrics is that the functional size of the software components is related to the duration and effort required for testing. Function point counts are a good reflection of the functionality; however, projects are often a mix of functional and non-functional requirements. In scenarios where the ratio of functional to non-functional is out of the ordinary, functional measure may not be useful.

Relative Measures. Measures in this category use the measurer's perspective as a framework to assess size. Story points and tee shirt sizing are examples of relative size measures. This form of measurement makes that same basic assumption that every other size category makes; that size is related to effort. The way the metric is used is typically different. Instead of using size to estimate how much effort a piece of work will require, relative measures are typically used to gauge how much work can be done in a fixed time by a fixed group. The development of relative measures engages the whole team, through techniques such as planning poker. The process helps the team determine size while learning about the user story being measured. Issues with this type of measure center on the need for the team to be stable or the need for reporting size for use in other measures.

Test case points are a hybrid approach. The method begins with test cases (a tangible thing) and then counts the steps, verification points, interfaces and factors in which baseline data is needed. These counts within a count are used to adjust the size of the test case in order to more closely relate size to effort. This approach uses concepts from the physical and functional approaches.

Each of these sizing approaches has pluses and minuses. When and where the value of these metrics outweighs the effort to generate these metrics is a subject of much debate as is which size measure or metric you choose. Where you land in this debate is heavily influenced by how you are organized, where you are in the value delivery food chain and whether the work is being done for a fee.

Which size metric makes sense for a testing organization is influenced by how testing is organized, where testing is incorporated into the value delivery chain and whether the work is being done for a fee.

Organization

How the people are organized to identify needs, develop software and deliver value will influence which sizing technique will be appropriate for testing. There are two basic competing models for organizing testers. The two models are independent test groups and testers embedded into the team. Variants of the latter include testers as part of the team and testers as matrixed members of the team. In an independent test model, developers complete their work (usually after unit testing) and then "throw" the work over the wall to the independent testers, whereas in the embedded version the work does not have to pass over a boundary.

Independent testing teams generally focus most of their efforts on planning and executing tests (these types of tests are often termed dynamic testing and include system testing through user acceptance testing). In this organizational scenario, testing teams size their work independently of the development team. Size is used to predict when work will be completed or how much work will be accepted by the team. Metrics that specifically leverage or count testing deliverables such as test cases or test case points are typically used.

Testers embedded in the team generally align to the same size metric as the development personnel (the decision of which size metric is often decided by the development personnel). Testers plan their work as part of the overall team or at worst, concurrently with development tasks. In this scenario, all three categories of size metrics are used. Examples of physical, functional and relative size metrics used include a number of requirements, IFPUG function points or story points.

Value chain

How testing activities, both dynamic and static (static testing includes techniques such as reviews and pair programming), are incorporated in the value delivery chain influence which size metric will be used to plan testing activities.

The more integrated into the development process testers are, the more likely the team will plan together using either a functional metric (function points) or a relative measure (tee shirt sizing or story points). Teams that are using any of the test first techniques (e.g. TDD, BDD and ATDD) are examples of scenarios in which testing is highly integrated into the value delivery chain. The higher the level of integration of testing into the development process, the more apt testers will be to leverage sizing metrics that reflect the ultimate functional deliverable versus counting individual physical items.

Similar to the scenarios in which testing is an independent group, when test activities are segregated to a separate phase, test teams often leverage physical size metrics (number of requirements or components) or hybrid sizing methods such as test case points.

Work for a fee

Outsourced testing is an extreme version of the independent test group. Outsourced test groups that price by project, face all of the same issues as teams doing any outsourced piece of work. With the exception of open, time and material contracts, the testing team needs some basis for the estimate that allows them to complete the work and make a healthy profit margin. Just like many development groups that have begun to leverage cost per function point, testing providers are experimenting with cost per test case point.

Conclusion

The size metric a test group leverages is rarely a random choice. Many of the influences are outside of the individual tester's span of control. Organization culture influences how testers are involved in the development process and whether they are segregated into separate teams. The best option for a size metric is the one that helps a team know how much work they can commit to completing and when that work can be completed well. ■

About the Author:

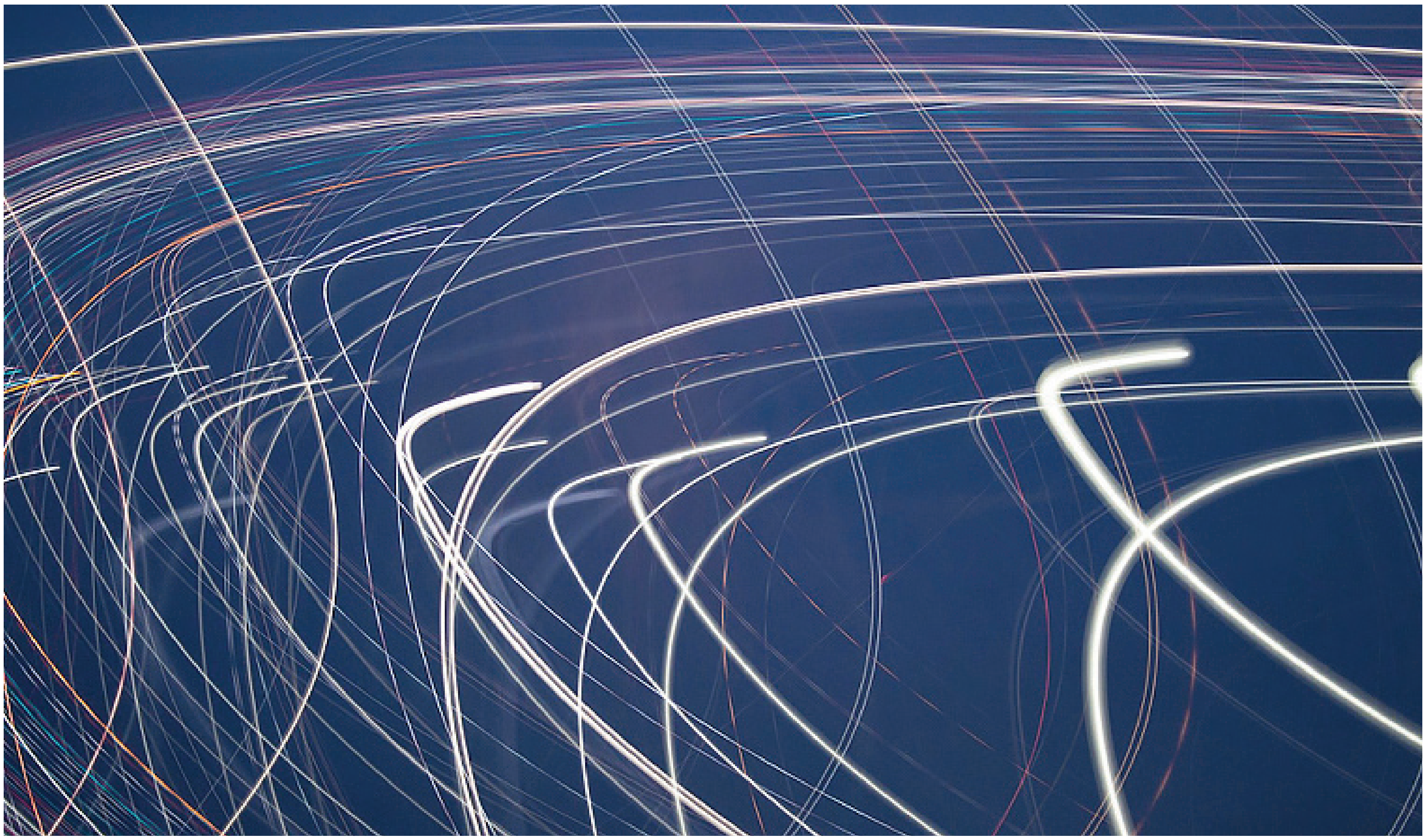
Tom Cagley *is a consultant, speaker, author and coach who leads organizations and teams to unlock their inherent greatness. He has developed estimation models and has supported organizations developing classic and Agile estimates. Tom helps teams and organizations improve cycle time, productivity, quality, morale and customer satisfaction and then prove it. He is an internationally respected blogger and podcaster for more than 11 years focusing on software process and measurement. His blog entries and podcasts have been listened to or read more than a million times. He co-authored Mastering Software Project Management: Best Practices, Tools and Techniques with Murali K. Chemuturi. Tom penned the chapter titled "Agile Estimation Using Functional Metrics" in The IFPUG Guide to IT and Software Measurement. His certifications include CFPS, IT-CMF Tier 2 Certified Associate, CSM, SAFe SPC, TMMi Assessor and TMMi*

ATTENTION

IFPUG is looking for volunteers to serve on some of the many exciting IFPUG committees. There are openings on the Communications and Marketing Committee, the Functional Sizing Standards Committee and the newly formed Partnerships & Events Committee. You can read the committee reports in this edition of *MetricViews* to learn more about each committee and the great work that is being done. Send an email to ifpug@ifpug.org and see how you can become a contributing member.

NON-FUNCTIONAL COMPLEXITY THRESHOLD IN FUNCTION POINT DELIVERY RATES (OR WHAT WE CAN LEARN FROM THE INSURANCE BUSINESS ON ESTIMATES)

By Carlos Eduardo Vazquez



What this article is about: How to improve productivity risk distribution among business areas and software development organizations; simplify work monitor and control by including non-functional measurement for functions with intrinsically high delivery rates due to implementation and design responses to non-functional requirements in organizations with mature IT governance practices.

When this article is useful: Function point estimates are great when average delivery rates are meaningful. However, it is not unusual to have some of the application functionality with outrageous high delivery rates. Since it is unknown how many functions hold up those poor performance numbers and how high they are, it is a risk management issue; and it adds up unnecessary variability to estimates now that we have IFPUG´s SNAP. You get the benefit of lower variability using a strategy like the insurance deductible without the overhead of measuring each functionality with SNAP.

What this article is for: The mean delivery rate estimation model is the simplest way to estimate from function points. Once a delivery rate is chosen by similarity, it is multiplied by the function points measured or approximated in order to plan or monitor work performance. Greater variability is the downside from this simple model. The variability is mostly due to “non-functional complexity” if work performance

"You should not use function points estimates for micromanagement of work packages addressing little functionality."

problems are spaced apart. The mean delivery rate tries to indirectly capture non-functional complexity; however, it is not normally distributed and is highly skewed. That is, there are few functions with extremely higher non-functional complexity. This proposal expresses non-functional complexity in terms of SNAP points per function point measured. It allows client and development organizations to establish an upper limit within the delivery rate, working as a threshold from which functions get to be measured with SNAP and the exceeding SNAP points are translated to a function point equivalent; in such a way the additional non-functional complexity contributes to the overall product size.

1. Introduction

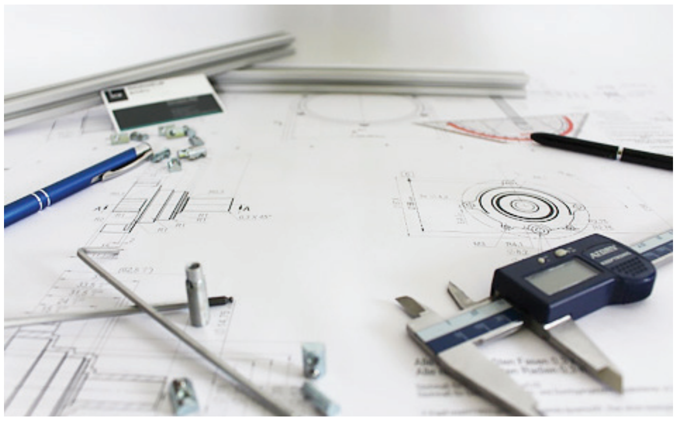
Function points measurement allows you to quantify a software application in terms of product units from a bird's-eye view, when compared to other software metrics which are primarily based on the product design and implementation. Software measurement from its functional requirements at the proper development level fosters users to better understand the measurement or approximation results as product units. It has a collateral effect of verifying if a set of requirements are mature enough to be prioritized to the next product increment development cycle, such as sprints if you use Scrum. I could spend days speaking about the benefits of function points and their role in sound work performance planning, monitor and control in the context of IT governance, specifically on the space among business areas or client companies and software development organizations.

As the character Ben Stark said in Game of Thrones, nothing someone says before the word "but" really counts. I believe it is not the case here, but despite all its benefits, there is a recurring issue when you have function points institutionalized in a corporation for estimate or prescribe software development effort or cost. Questions, such as how to estimate a "complex" process, rise. In fact, if you look closer at the question, you can depict a complaint. The question is a complaint about an insufficient amount of budget or time resulting from the multiplication of the function points measured or approximated to the target delivery rate. Questions of the same nature rise on the opposite direction as well, such as, why so many hours for a drop-down list box.

2. It is (not) normal

When you land from your bird's-eye view to the level of a specific function, actual delivery rates variability is huge. This variability is not necessarily due to work performance issues. Since function points prioritize coverage over depth and ignore design and implementation decisions, delivery rates indirectly absorb the impacts on cost and effort in its averages values. That's why you should not use function points estimates for micromanagement of work packages addressing little functionality[1].

When you climb higher back to your bird's-eye view, mean delivery rates become sound again due to the balance between functionalities underestimated and overestimated. Paul Below[2] pointed out that "metrics tend to have skewed distributions, and relationships tend to be non-linear." The use of a projected mean delivery rate in such a context is tricky.



In 2008, my company assisted IBM on validating and drafting a new outsourcing agreement based on function points as product unit to prescribe work orders effort. There was one request I was not able to provide at the time.

In such an agreement, the target delivery rate is a risk management decision from both parties. First, the delivery rate is determined based on conditions prior to the change. For instance, prior to the change, some work orders wouldn´t be issued because of the high cost to implement some non-functional requirements. Afterwards, that work order effort will be prescribed based on functional requirements; it will be no obstacle to prioritize the request and the work order is issued. Second, the target delivery rate seldom is determined using non-linear transformations; when it is the case, the models derived from such transformations are more complex and less intuitive to understand.

The request was to establish something like an insurance deductible policy, in such a way the target delivery rate would apply up to a certain effort level derived from non-functional requirements and, from that point on, IBM would have additional units measured and additional effort and budget assigned to the work order. However, there was no Software Non-functional Assessment Process (SNAP) as of 2008.

3. SNAP

I will assume the reader has a basic knowledge of SNAP, its relationship to function point measurement and its role in measuring components, processes or activities that are used in order to meet the non-functional requirement. IFPUG released the first draft version of SNAP less than one year later in 2009[3]. At first, I was a bit skeptical due to its design; but it is not so different from function points design[4] and I´ve been using it successfully for more than 30 years.

4. The client organization

Ten years later in 2018, after considerable evolution of the assessment process, another client organization presented my company with a similar problem suited for applying SNAP as part of the solution. The client organization has reached a high IT governance maturity and uses function points and different classes of target delivery rates. It uses functional size approximation in conjunction to these delivery rates to manage supplier performance, establish budgets, negotiate strategic corporate alliances; prescribe and eventually negotiate individual project budgets.

The client organization decided to conduct a SNAP Proof of Concept (PoC) as a critical solution component to address the variability and risk issues discussed in so far. These issues can be summarized in complaints from suppliers about a "complexity felt, but not measured" in the client organization regarding some functionalities. In order to address it, the PoC measured and analyzed data at the functionality level instead of the project level.

The client organization established the target delivery rate should apply as a rule in general, and exceptions should have a different treatment assigning, somehow, greater volume of product units regarding this exceptional complexity. Furthermore, these exceptions should not promote additional measurement effort for the other common functionalities. It also established the artifacts used as input to the non-functional assessment must be the same available as when the functional measurement happens.

5. Non-functional density

CPFS and CSP from two different organizations measured 209 functionalities, totaling 932 Function Points and 4,867

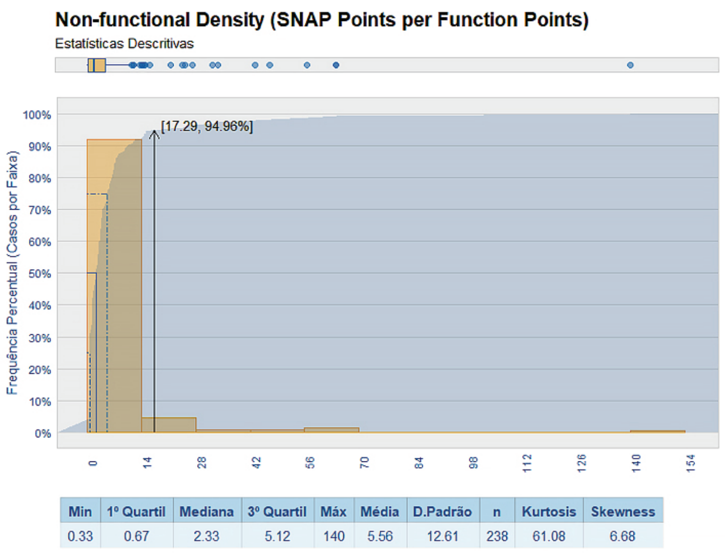
"It is not possible to model the effort or cost depending on the functional and non-functional measurement in the context of the client organization."

SNAP Points. The project team defined the ratio of SNAP Points per Function Points as Non-functional Density and used it to quantify the "complexity felt, but not (yet) measured." The PoC requirements led the project team to select 7 from 14 SNAP subcategories in the assessment; table 1 lists them.

Sub-category	Description
1.1	Data Entry Validations
1.2	Logical and Mathematical Operations
1.3	Data Formatting
2.3	Multiple Input Methods
2.4	Multiple Output Methods
3.2	Database Technology
3.3	Batch Processes

Table 1 – List of SNAP subcategories considered

The PoC results led, later, to a SNAP Pilot Project over a product increment with some issues about the target delivery rate. It added 29 transactions and 200 FP to the original PoC data. Figure 1 summarizes the non-functional density distribution from the combined data from the PoC and the Pilot.



Min	1º Quartil	Mediana	3º Quartil	Máx	Média	D.Padrão	n	Kurtosis	Skewness
0.33	0.67	2.33	5.12	140	5.56	12.61	238	61.08	6.68

Figure 1 – Non-functional complexity distribution.

The combined distribution verified after the pilot was like the distribution verified on the PoC. Both indicate outlier functions with higher non-functional density.

6. Determine the deductible threshold

To determine an exception to the general is a necessary step toward the solution. An exception is a kind of outlier and there is no rigid mathematical definition for what constitutes an outlier. Determining whether an observation is an outlier is ultimately a subjective exercise[5]. The usual criteria to identify outliers, used in the boxplot depicted in Figure 1, is not adequate for the data is highly skewed. The project team decided to use an adjusted boxplot for skewed distributions[6]. The result set 25.72 SP/FP as the threshold, from which an outlier should be identified.

The result places 4% of the transactions assessed as exceptions. The client organization with the project team support adjusted the threshold to 17. 27 SP/FP in a decision to consider 5% of the greatest non-functional density transactions as exceptions. So, the target delivery rate, that is the quantity of staff-hours prescribed per function point, would comprise up to 17.27 SP/PF of non-functional density. Transactions with non-functional density beyond this threshold should accrue additional units due to its exceptional profile. Figure 2 shows the non-functional density from the study compared to the normal distribution and the defined threshold; functionalities beyond it accrue additional measurement.

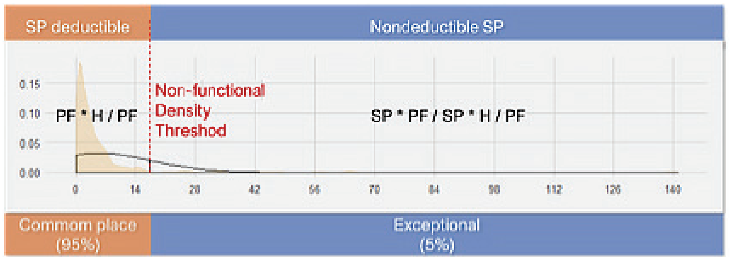


Figure 2 – The threshold in the non-functional density context

The idea is to foster a healthy relationship between the client and software development organizations by better differentiating poor work performance from part of the effort or cost incurred, but not included in the time or budget prescribed by the functional measurement and the average target delivery rate due to exceptional non-functional requirements.

7. Non-functional accrual

To determine the non-functional accrual for the functionalities with non-functional complexity beyond the threshold, it is necessary to map its additional SNAP points to a unit equivalent to function point, in such a way the value is soundly aggregated to the original function points measured (the non-deductible area in figure 2). That is, to answer the question, "In general, how many function points correspond to a SNAP point?"

It is not possible to model the effort or cost depending on the functional and non-functional measurement in the context of the client organization. The actual effort is completely dependent from the function point measurement.

The project team excluded the functionalities with non-functional density beyond the threshold, because the intent is to derive a relationship between SNAP points and function points measured in general and not in exceptional functionalities.

Due to data skewness and data transformation required to model function points as a dependent variable from SNAP points, the best suited formula derived from linear regression using the ordinary least square (OLS) method includes an exponential component. The client organization requested an alternative solution.

The project team decided to use a robust regression. Robust regression is an alternative to least squares regression when data is contaminated with outliers or influential observations[5]. The result was the coefficient of 0.1422 FP/SP to determine a function point equivalent product unit representation of the non-functional surplus.

8. Challenge accepted

As stated earlier, the idea is to trigger a SNAP measurement only in exceptional cases in order to minimize measurement overhead. The development organization facing a functionality with a "complexity felt, but not (yet) measured," challenges the client organization if SNAP measurement confirms the higher than the threshold non-functional density.

For example, the development organization measures a 6 FP External Entry 386 SP, thus the non-functional density is determined as 64,33 SP/FP; beyond the 17.27 SP/FP threshold. The deductible fraction already included in the target delivery rate is up to 103.62 SP (6 FP x 17.27 SP/FP). There is a 282.38 SP surplus when the SNAP measurement is subtracted from the deductible fraction (386 SP – 103.62 SP).

Once the client organization validates the challenge and the measurements against the requirements, an additional function point equivalent allows to capture this surplus due to non-functional requirements in the overall product size.

For example, due to the higher non-functional density verified, the External Entry functionality accrues an additional 40 PF equivalent (282.38 SP x 0.1422 PF/SP) to the original 6 PF measured.

9. Conclusion

Although the results presented are not universally applicable, mostly for a specific set of SNAP categories were selected due to project requirements and the bias introduced by single organization data, the idea of a non-functional density threshold proves to be a sound concept with a viable implementation.

The world is in the middle of the digital transformation phenomenon. Not seldom, companies are willing to neglect operational efficiency and governance excellence in a tradeoff for agility as a survival strategy. Software measurement must be simple and preferably a byproduct of the development itself. To restrict the SNAP measurement effort to 5% of the transactions tends to reduce the resistance to its adoption; furthermore, to measure the non-functional surplus of these exceptional functionalities tends to leverage function point business models adoption and provide assertive answers to some of its more common criticisms. ■

References

[1] VAZQUEZ, C. SIMÔES, G. (2013). Análise de Pontos de Função: Medição, estimativas e gerenciamento de projetos de software. 13th ed. Érica.

[2] IFPUG (2012). The IFPUG guide to IT and Software Measurement. New York: CRC Press.

[3] IFPUG (2017). Software Non-functional Assessment Process (SNAP) Assessment Practices Manual. Release 2.4.

[4] IBM Corporation (1979). Measuring Application Development Productivity. [online] Monterey, CA: Joint Share, GUIDE. Available at: http://www.fattocs.com/files/pt/artigos/MeasuringAplicationDevelopmentProductivity.pdf [Accessed 29 Nov. 2019].

[5] Rousseeuw, P. and Leroy, A. (2005). Robust Regression and Outlier Detection. 4th ed. Hoboken: John Wiley & Sons.

[6] M.Huberta, E.Vandervierenb, "An adjusted boxplot for skewed distributions" Computational Statistics & Data Analysis, Volume 52, Issue 12, 15, August 2008, Pages 5186-5201.

About the author:

***Carlos Eduardo Vazquez** has more than 25 years of experience in software development, maintenance and management. He has been a function point analysis user (1991), instructor (1993), CFPS (1996) and CSP (2018) by IFPUG and COSMIC (2012); among the first Brazilians to hold them. He coauthored the book APF:* Medição, Estimativas e Gerenciamento de Projetos de Software, *currently in its 13th edition. He coauthored the book* Engenharia de Requisitos: Software Orientado ao Negócio *and became certified by the IREB (2016). Primarily a business-oriented professional, he founded FATTO Consulting, where coordinates research and development for consulting services.*

Measurement Origins, When and Then, Provoking Us to Think Again

By Joe Schofield

Leadership often talks about software quality, but rarely invThe International Function Point Users Group (IFPUG) was formally organized in 1986, in Naperville, Illinois, U.S. The development of Function Point Analysis (FPA), based on Allan Albrecht's seminal work at IBM, became the foundation for the measurement of software based on its functional size. Today, through IFPUG, FPA has grown with a worldwide membership, professional certifications, international conferences, the emergence of SNAP[1] and supporting standards.[2,3] The popularity of FPA has spawned several other function point and function point-like counting methods.

Software measurement remains a daunting task. What do we measure? Under what conditions, without introducing a Hawthorne effect, can we measure? How do we collect those measures without the introduction of bias? How and where do we store measures? How long are those measures useful? What measures are comparable? How do we ensure we are measuring the right "things?" Are our measurements better than those we'd find from a random number generator? Who should collect the measures? Who is qualified to take measurements?[4] Do we want to divert energy from "real work" to feed values into tools? How long until we should expect to see "improvement?" How do other dynamics in the environment, business, process and organization impact what we measure and how those measurements are perceived?

Some of my own attempts to explore and invigorate the measurement discussion include: comparing lines of code to a random count of ants,[5] statistical analysis demonstrating that no competent team member could be predicted to write "better" or "worse" software in a diverse environment,[6] analyzing function points with use case points and story points,[7] Agile quality measurements,[8] tips on how to cheat with Agile measurements,[9] defect reduction through capture-recapture statistical processes[10] and even challenging how we employ six sigma techniques.[11]

With such a girth of content already written by numerous scholars, what's left to cover? Fortunately, or not, plenty remains to be examined. Thanks to tools in general, and software management tools in particular, we have way too much measurement data today[12] and it's only going to get worse. The world's insatiable appetite for data grows unabated. Direct feeding from team members and the automated collection of process data and performance measures are time consuming and often indigestible by the human mind alone. Distilling vast data dumps into meaningful insights is elusive as the data itself morphs and the sources increase faster than the thoughtful mind can reconfigure. Data centers are built for storing unprecedented volumes of bits and "the cloud" expands to accommodate a runaway collection of digital data awaiting its mining by analytics engines. We assemble the data in tables, present it in an array of charts, sort it, filter it, render it, plot it, trend it, predict it, balance it in scorecards and sometimes just ignore it. Professional sports teams play by it; nations govern by it; organizations determine their future by it; markets react to it. One might ask, who started all this measurement mania?

> "Distilling vast data dumps into meaningful insights is elusive as the data itself morphs and the sources increase faster than the thoughtful mind can reconfigure."

Traces of the roots of measurement can be found in historical documents. One example is, "In the beginning…" so starts Genesis, "…God created the heavens and the earth." Reading on we note six days of creation and a day of rest. Aha! God started the counting; therefore, the blame surely rests with the Creator. While some think this conclusion settled, a more careful review may be warranted.

In the southern parts of Africa, the lower leg bone of a baboon may be the first measurement recording instrument. Some accounts date the Lebombo bone at more than 40,000 years.[13] Notches on the bone are depicted as tracking the lunar cycle and menstrual cycles—maybe even used to first describe the notion of being "late" in the start of a cycle, as opposed to being late on project milestone, a more modern and often, consequential usage.

The use of one's fingers has been suggested as the most likely first method of counting. Sexagesimal numerals were used in Babylonia to count time and angles[14]—thus the 60-minute hour and the 360 directional degrees with which we are familiar today. Perhaps "late" could be defined with more precision than the passing of a day on a primate's leg bone. Even today we tend to conflate precision and accuracy. Better to know that the occurrence of an event is a "notch" (day) late than to pretend that the same event is 2,880 minutes (48 hours) delayed and miss it by an entire notch.

The tracking of units of time addresses the first of the three "triple" constraints of time, cost and scope often used in project management today. Quality is also considered a constraint in many industries. Cost too originates in our ancient past. Excavated coins from circa 800 BC were discovered in the Temple of Artemis.[15] Between about 1000 and 600 BC dynasties and kingdoms in India are suspected of minting silver and copper coins.[16] Around 600 BC in what is today Turkey, Lydia's King Alyattes established what is believed to be the first currency. Due to the weight of hauling coinage, the Chinese are thought to be the first to use paper currency. Curiously, the Incas had no notion of a currency. In the 1600s furs were the preferred currency of Siberia. Yet, copper, silver and gold coins; paper "bills," electronic funds transfer (1800s), plastic (1940s), mobile (circa 2000), wearables and bits each played an evolutionary role in commerce and trade.[17] Bartering preceded all of these forms of currency and is still recognized as taxable in the U.S. by the Internal Revenue Service (IRS).[18]

The measurement of cost and schedule can certainly be traced back centuries if not millennia. What about the third dimension—scope? The evidence of ancient payments for bit and bytes is scant, actually, non-existent. But scope and size were often measured by weights and mass, using scales and pottery (though some may prefer the word "containers" today).

"Even today we tend to conflate precision and accuracy."

Body parts were used to measure length: elbow to middle fingertip (cubit), hands and fingers in biblical times. The Greeks and Romans inherited the foot from the Egyptians and divided it into 12 inches. One thousand paces (mille passus) became a mile, a yard became 36 inches or the size of a man's waste line—definitely not to be considered medical advice. Pottery measured liquids and grains from 2 to 26 liters.[19]

Today we measure sea vessels by gross displacement tonnage, oil with barrels, natural gas with cubic feet, water in millions of gallons per second and acreage feet, gallons, liters and more; distance in light years, perfumes in ounces and illicit substances in grams. Obviously and intentionally, the preceding measures are but an abridged sample.

What if the measure of software, and related products and services were just as simple, and precise? Much as we've seen with the original FPA, innovators and differentiators would intervene to disrupt markets and the status quo. Marketing campaigns would petition C-level leadership that "there's a better way."

In the sustained history of change in the measurement world, different and more is sometimes valued over better, cheaper and less. Be wary of ongoing improvement cycles that aren't self-sustaining, objective, meaningful and minimal. For success with measurement systems, consider outcomes over outputs, impact over impactful, needs over haves and progress over change.

No doubt IFPUG will continue to play a leading role in attempts to quantify software capability. Like a diet or belt-tightening in a new year, can we survive with less? 🟧

References

1 https://www.ifpug.org/about-snap/ retrieved 12/28/2019

2 https://www.iso.org/standard/51717.html retrieved 12/28/2019

3 https://standards.ieee.org/standard/2430-2019.html retrieved 12/28/2019

4 Why You Need a Certified Function Point Specialist (CFPS); Joe Schofield; MetricViews; International Function Points Users Group; January, 2013

5 Lessons from the Ant Hill - What Ants and Software Engineers Have in Common; Joe Schofield; Information Systems Management; Winter 2003

6 Function Points, Use Case Points, Story Points: Observations from a Case Study; Joe Schofield, et al; CrossTalk; May / June, 2013

7 Counting Lines of Code: Virtually Worthless for Estimating and Software Sizing; Joe Schofield; IT Metrics and Productivity Journal; December, 2009

8 Considerations for Establishing Agile Quality Metrics; Joe Schofield; MetricViews; September, 2019

9 Inflate Gate: Mastering Overestimation for Agile Software Projects; Joe Schofield; Computer Aid's Accelerated IT Success; (Featured Article); IT Metrics & Productivity Institute; August, 2015

10 The IFPUG Guide to IT and Software Measurement; Joe Schofield; IFPUG; 2012; Chapter 36

11 When Did Six Sigma Stop Being a Statistical Measure?; Joe Schofield; CrossTalk; April, 2006

12 Measurements in the IOT; Joe Schofield; IFPUG International Software Measurement & Analysis (ISMA14) Conference; Cleveland (USA); September 15, 2017

13 https://en.wikipedia.org/wiki/Lebombo_bone; retrieved 12/28/2019

14 https://en.wikipedia.org/wiki/History_of_ancient_numeral_systems retrieved 12/28/2019

15 https://en.wikipedia.org/wiki/History_of_coins retrieved 12/28/2019

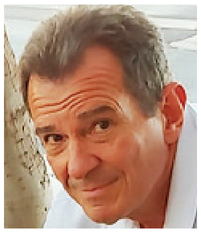
16 https://en.wikipedia.org/wiki/Coinage_of_India retrieved 12/28/2019

17 https://www.telegraph.co.uk/finance/businessclub/money/11174013/The-history-of-money-from-barter-to-bitcoin.html retrieved 12/28/2019

18 IRS, Form 1099-B, Proceeds from Broker and Barter Exchange Transactions

19 https://en.wikipedia.org/wiki/History_of_measurement

About the Author:



Joe Schofield *is a Past President of the International Function Point Users Group. He retired from Sandia National Laboratories as a Distinguished Member of the Technical Staff after a 31-year career. During 12 of those years he served as the SEPG Chair for an organization of about 400 personnel which was awarded a SW-CMM® Level 3 in 2005. He continued in that role to CMMI® Level 4 until his departure. Joe holds eight Agile-related certifications: SA, SCT™, SSMC™, SSPOC™, SMC™, SDC™, SPOC™, and SAMC™. He is also a Certified Software Quality Analyst and a Certified Software Measurement Specialist. Joe was a CMMI Institute certified Instructor for the Introduction to the CMMI®, a Certified Function Point Counting Specialist and a Lockheed Martin certified Lean Six Sigma Black Belt. He completed his master's degree in MIS at the University of Arizona in 1980.*

Certification Committee

The incoming chairperson, Mahesh Ananthakrishnan and the incoming vice chairperson, Cinzia Ferrero would like to thank our past chairman Gregory Allen for his outstanding contributions and support to the Certification Committee.

Ananthakrishnan is an information technology expert, with a proven track record of project and program management delivery for more than two decades. He brings a global perspective, having played multiple roles as delivery manager, PMO leader and estimation CoE leader in the United States, UK and India. He currently works as an estimation and deliverability assessment lead in Cognizant technology solutions. He has served on the IFPUG Certification Committee since 2004.

Ferrero is a member of the IFPUG Certification Committee, a CSP since 2017, CFPS since 2018 and CCFL since 2019. She is a software measurement and process improvement specialist at CSI-Piemonte (the Information System Consortium to which Italian Public Administration entrusts the management and implementation of its ICT services) in Turin, Italy. She has more than 20 years of experience in information technology, supporting Italian Public Administration in state and local government (e-commerce, web communication, professional training, software metrics).

Exam Updates

With the help of the new exam partner, Brightest, the Certification Committee has launched a CSP exam in English. The CFPS exam is already available on the Brightest platform in English, Italian and Brazilian Portuguese. ■

Communications and Marketing Committee

By Antonio Ferre Albero, Committee Chair

Many things have happened since the previous *MetricViews* edition: the IFPUG SNAP method has become an IEEE worldwide standard; the IFPUG Annual Meeting; IFPUG 2019 elections (welcome Filippo De Carli, and thanks for all your years on the IFPUG board to Tom Cagley: it will be great joy for us if he remains close to IFPUG); IFPUG has changed the exam provider (to Brightest Platform); participated in the Brazil Métricas 2019 conference; a face-to-face meeting of IFPUG board members and chairs in Baltimore, U.S., in October; IFPUG has a new president (Christine Green) who replaces Mauricio Aguiar after different years of service as president. This type of information is communicated by the Communications and Marketing Committee (CMC) through a variety of channels including the IFPUG website and announcements.

Another important asset of the IFPUG CMC is *MetricViews*, the publication that just now you have in your hands; the metrics window to the world with a universal and plural collaboration.

In CMC we have many initiatives. But for putting in place all that is needed are volunteers, not only initiatives and ideas. The IFPUG activity is done by volunteers and now CMC needs volunteers to put in place all the initiatives. We need metrics-passionate persons. Perhaps you? Let me reuse the words of Christine Green, the IFPUG president, recalling the importance of the volunteer concept "IFPUG is run primarily by volunteers. Volunteering in IFPUG is for many a passion and we all seem to have the common goal to improve the success of software projects. Only by volunteering, an organization like IFPUG can strive to accelerate and grow." Are you interested in being a passionate of IFPUG working as volunteer in the CMC? You can contact ifpug@ifpug.org. ■

Functional Sizing Standards Committee

By Daniel French, Committee Chair

As we enter the New Year, there are two key projects that the Functional Sizing Standards Committee (FSSC) is participating in. The first is the Marymount University study conducted by Charley Tichenor, Esteban Sanchez and Micheline Al Harrack. The study analyzed the General Systems Characteristics (GSCs) and made recommendations to both the FSSC and the Non-Functional Sizing Standards Committee (NFSSC). The purpose of the study was to analyze the GSCs and determine if they need to be updated, deleted or removed completely from the Counting Practices Manual (CPM) as well as assessing what changes need to be made to the SNAP Assessment Practices Manual (APM). There were several excellent suggestions made in the study, but there will be no changes to the CPM. The NFSSC will be working further with the study team to identify potential updates to the APM.

The second major project underway is the acquisition of the Simple Function Point methodology by IFPUG (SiFP). A task force has been assembled, led by Vice President Chuck Wesolowski, to determine how the SiFP will be incorporated into the IFPUG FP methodology. Members of the FSSC, NFSSC and Roberto Meli, developer of the methodology, are part of this important task force.

The committee continues to meet monthly and has recently completed the updates to the Case Study. We are also working on Application Boundary, Elementary Process and Mobile Application white papers.

We have received applications from several candidates to fill the open positions currently on the committee. During the coming months, the FSSC will be reviewing the applications and conducting interviews prior to making a selection.

The committee appreciates the support of the IFPUG membership and is always looking for new projects to work on. Some topics under consideration for our next projects include MicroServices, Agile and Cloud. We welcome suggestions from members on topics of interest. Please submit your suggestions to ifpug@ifpug.org. ■

Industry Standards Committee

By Carol Dekkers, Committee Chair

The Industry Standards Committee (Carol Dekkers, Steve Woodward, Talmon Ben-Cnaan) was named the IFPUG Committee of the Year by the IFPUG Board of Directors. We are honored to serve our membership.

The IFPUG Industry Standards Committee continues to work on your behalf to advance IFPUG measurement standards to an international level. With the recent board of directors' reorganization, our committee scope is changing and morphing (for example the ISBSG liaison is moving to the new Strategic Partnership committee.)

The standards committee continues collaborating with several IT standards development groups, this includes IEEE and ICEAA, to highlight the benefits from applying software sizing methods. Software metrics are extremely important competencies to mature in the new "trust but verify" service consumption models.

IFPUG continues to be a permanent voting member of the U.S. Technical Advisory Group (TAG) to ISO/IEC JTC1 SC7 and as a result we attend three domestic U.S. meetings per year (two are virtual, one is onsite) and help to formulate and vote on U.S. positions for international standards under development. Carol Dekkers represents us at the TAG meetings/conference calls.

Steve Woodward is a member of the Canadian standards delegation to ISO/IEC JTC1 SC7 and is active within a number of subcommittees including the WG6 SQUARE series of quality standards. He and Carol can represent IFPUG internationally in its second role as a Category C Liaison to SC7.

Talmon Ben-Cnaan (chair of the NFSSC committee) was successful in chairing an IEEE computer society standards working group that recently published IEEE 2430 SNAP. This was a multiple month-long project to standardize the IFPUG SNAP (Software Non-functional Assessment Process) for non-functional software sizing.

If you'd like more information about any of the work underway or would like to volunteer to assist with the Industry Standards Committee, please reach out to us by email or via the IFPUG office. ■

International Membership Committee

By Saurabh Saxena, Committee Chair

The International Membership Committee (IMC) has always acted as a bridge between IFPUG and its members. This year we saw Dacil Castelo get re-elected to the IFPUG board and continue as the board liaison of IMC. Two of our members, Gianfranco Lanza (Italy) and Rajesh Koduru (India), are no longer part of the IMC and new Italy country representative Giovanni D'Alessandro has joined us. We thank Gianfranco and Rajesh and welcome Giovanni in the new role.

In our continued efforts to reduce pain areas of IFPUG and its members, we are working on the following:

· Simplifying the volunteer form and process

· Collaborating with 'Brightest' for the smooth transition of CFPS/CFPP exams to the new provider

· Creating FP and SNAP awareness sessions in universities and colleges

We are committed to enhancing the IFPUG membership experience and providing the first line of contact for all IFPUG related queries across globe. ■

Non-Functional Sizing Standards Committee

By Talmon Ben-Cnaan

The Non-Functional Sizing Standards Committee welcomes two new volunteers, Nicolantonio Auciello and Fabrizio Di Cola.

SNAP as an IEEE Standard

SNAP is now an international sizing standard: IEEE 2439-2019.

SNAP manual translation

SNAP 2.4 translation to Brazilian Portuguese and to Italian will be available soon.

SNAP and GSCs

Accurate effort estimation needs to have the functional size, the non-functional size and the impact of the project's requirements and constraints. Project requirements and constraints do not add size, but they affect the effort.

Before SNAP was available, the GSCs were used to compensate for the non-functional part and the project's requirements and constraints. Therefore, SNAP and the GSCs somewhat overlap. A white paper will guide users how to separate the non-functional part so that FPA, SNAP and revised GSCs will provide better information needed for estimation. ■

Partnerships and Events Committee

By Sushmitha Anantha

We have a new name and redefined mission.

This committee was earlier known as the Conference and Education Committee (CEC), and is now the Partnerships & Events Committee (PEC). We are glad to expand our horizons by focusing on new strategic partnerships to discover opportunities to share software metrics methodology worldwide.

In 2019, as CEC, we had successfully organized ISMA^{17} in Bangalore (India) and we worked to renew the committee activities in order to try to create value for our members trough events around the world.

Currently, the committee is focusing on:

- Planning and participating in a Quality Conference of Pan-Asia Importance in Malaysia in September 2020.
- Identifying new strategic partnerships and taking them towards workable agreements.
- Working on growth of the PEC with induction of new members.

The PEC is delighted to work with anyone interested in helping us. Would you like to join the PEC? Send an email to ifpug@ifpug.org or complete the volunteer form (Select Conference and Education Committee until the volunteer form is updated with the new committee name) available on the IFPUG website. ■

STAY CONNECTED

As an IFPUG member, you are part of an international association dedicated to improving the quality and future of the information technology industry. Are you taking full advantage of all that your membership offers?

Benefits include:

- Access to education and professional growth through semi-annual IFPUG Workshops and the annual IFPUG Conference at special member rates.
- Opportunity to join a local IFPUG Chapter, where you can exchange ideas, share experiences, and learn about new techniques on an ongoing basis, in your area.
- Participation in IFPUG communities to advance state-of-the-art software measurement and professional networking with colleagues from around the world.
- Professional certifications, which establish your credentials as a specialist in the growing field of software metrics.
- Access to state-of-the-art products and services at vendor showcases during the annual conference.
- Special member rates on IFPUG materials (e.g., the Function Point Counting Practices Manual and the International Software Benchmarking Standards Group publications).

IFPUG's social media channels allow you to stay connected to your fellow IFPUG colleagues and the HQ staff.

Be Informed! Stay Connected!

Advertise Around the World with IFPUG

Argentina	China	France	Italy	Netherlands	Spain	United Kingdom
Belgium	Colombia	Ghana	Japan	New Zealand	Sweden	United States
Brazil	Croatia	Hong Kong	South Korea	Peru	Switzerland	
Canada	Czech Republic	India	Luxembourg	Poland	Taiwan	
Chile	Denmark	Israel	Malaysia	Singapore	Thailand	

Spread your message to a global audience in 32 countries across 6 continents.
Promote your product or service by placing a highly visible ad in the September 2020 issue of *MetricViews!*

Contact IFPUG Headquarters at +1-609-799-4900 or ifpug@ifpug.org.