February 2016 Vol. 10 • Issue 1

INTERNATIONAL
FUNCTION POINT
USERS GROUP
191 Clarksville Road
Princeton Junction, NJ 08550
USA

MetricViews

www.ifpug.org

SHARING Experience

Inside...

In This Edition 2
Letter from the President 3
From the Editor 3
ISMA12 4
Vendors' World 4
High Efficiency Defect Removal for Software Projects 5
Thinking Outside the Box Using FPA on Emerging Technologies 8
What's the Story (points) with Function Points 11
Consistency is King: Counting FP for Agile/Iterative Software Development Projects 13
Boundaries & Partitions in SOA Architectures 18
Committee Reports
- Certification Committee 20
- Communications and Marketing Committee 20
- Conference and Education Committee 20
- Functional Sizing Standards Committee 21
- Non-Functional Sizing Standards Committee 21
- International Membership Committee 22
- ISO Standards Committee 22

Past Presidents Message 22
Behind the Scenes 22
Documenting the Functional Size Measurement 24
ISMA11 in Brazil 27
Board of Directors 28
Committee Rosters 28
CFPS Certification Listing 29
CFPP Certification Listing 30
CSP Certification Listing 31

In This Edition

Some really GOOD articles in this edition of *Metrics Views*.

We start with some sound advice and information from one of the key voices in metrics over many years. Capers Jones discusses some metrics that have long term impact and have been proven to be a useful focus for improvement.

Close practical involvement with sizing for $/FP related contracts always leads to an intense examination of sizing approaches and paradigms—hence the Brazilian experience reflected in "Thinking Outside the Box". This flexibility of thinking and understanding of the basis of function point sizing is also illustrated in two articles on Agile processes and associated measurements. Both of these articles demonstrate the depth of experience applied to the task and the issues encountered and addressed. Well worth reading.

We also review some of the difficulties associated with defining boundaries with modern architectures—and demonstrate how SNAP may assist in more accurately sizing a requirement.

And some important basics—what you really need to document in order to be able to fully use and maintain a function point count (although examples use a basic spreadsheet, there are commercial tools that offer considerable assistance).

And remember—if you wish to comment on or review any article, you can do so at the IFPUG website.

Message from the President

Tom Cagley

LETTER FROM THE PRESIDENT

In 1986, Bill Huffschmidt was elected the first president of IFPUG. In 1986, IFPUG was a nascent organization that, thirty years later, has matured into an organization supporting two sizing standards, a range of certifications and myriad publications. Over the years, our membership has grown. The thirty years since the founding of IFPUG have been marked by a lot of growth and change.

However, IFPUG's job is not done. For example, we are still early in the growth (albeit accelerating) of the Software Non-Functional Assessment Process (SNAP) sizing standard. The Non-functional Sizing Standards Committee has a long backlog of ideas for tweaks to the method, papers, and content to be developed. Every committee has a similar backlog. Many of the committees are looking for help. Please complete a volunteer form, and then reach out to the chairs of the committee or committees you are interested in becoming involved with. If you are not interested in joining a committee but have an idea that will help IFPUG grow please reach out to me at ifpug@ifpug.org. I will be happy to listen to your ideas, thoughts or complaints.

Thirty years after Bill became IFPUG's first president, I am looking forward to serving my second term. IFPUG is different now. In conversations with members recently in Krakow, Poland I was reminded that IFPUG has the most value when members interact with other members. Real interaction is more than presentation and letters. Interaction has to be synchronous; people talking to other people. In January, I will begin hosting monthly open forums for members to attend. These will be held at local noon in the time zones of the largest IFPUG communities. I will publish a schedule in early January. The goal is to talk and have some fun.

Bill Huffschmidt was the instructor for my first Function Point 101 class. I still remember that he wrote the word function points on the board and asked what the first three letters spelled. I would like you to get involved and I will do my best to make IFPUG and function points fun!

From the Editor's Desk

Paul Radford

Sharing Experience

In this issue of Metrics Views, you can almost feel the experience behind the words.

One of the core reasons for the existence of IFPUG is to enable and facilitate the sharing of experience in the business or process of applying software metrics. Whilst rules and guidelines are a key basis for measurement, the concepts of function point analysis are applicable to all software capability and thus require consistency and relevance for many technologies and many masters.

However, what sounds good in principle is sometimes startlingly poor in practice. The experience of those who have applied theory and learnt from it, revised again and reviewed intelligently – these stories, these contacts are a key part of what IFPUG is all about.

And the need for software measurement has never been more important.

Software has been outsourced. In many different ways. And places. With different laws. And different tax regimes. Understanding the real comparative cost of multi outsourced software providers is a complex task. Understanding the relationship of costs to functional deliveries has never been so important.

On the other hand, the cost of software is a flexible commodity that can quickly move profits from one company division to another; from one company to another; from one country (tax regime) to another. Without appropriate measurement, software costs can be as much or as little as is convenient.

Whether software measurement (and benchmarking) has an integral part to play depends to a large degree on the attitude of business, lawmakers and auditors.

But our experience and our data make useful analysis possible. For those who want to know.

Perhaps we should let them know.

MetricViews

Published twice a year by the International Function Point Users Group (IFPUG), headquartered in Princeton Junction, New Jersey, U.S.A.

***MetricViews* Winter 2016**

Editor
Paul Radford

Assistant Editor
David Herron

Views and opinions presented in *MetricViews* articles may not represent those of the International Function Point Users Group (IFPUG).

Please submit all articles, news releases and advertising to:

IFPUG/*MetricViews*
191 Clarksville Road
Princeton Junction, NJ 08550
(609) 799-4900
ifpug@ifpug.org

The 12th IFPUG International Software Measurement & Analysis Conference

"Creating Value from Measurement"

May 3-5, 2016 – Rome, Italy

Coming back to Italy after 20 years, this new edition of the IFPUG ISMA Conference will provide a forum for practitioners and researchers to discuss most recent advances in planning and sustaining measurement programs from both practical and theoretical perspectives in the scope of software value creation and value-based management in software product and service development organizations. We invite professionals responsible for, involved in, or interested in software measurement to share innovative ideas, experiences, and concerns within this scope.

ISMA 12 is organized by:

GUFPI ISMA

on behalf of:

Vendor's World

DCG Software Value

Pennsylvania, USA

DCG Software Value (formerly David Consulting Group) is a global provider of software analytics, software quality management and Agile development services. As the industry's leading independent provider of software sizing, we offer both Function Point Analysis and SNAP sizing services.

Our mission is to help IT and the business to collectively visualize and discuss the value of software development in order to foster improved decision making and resource management and to quantifiably impact a company's bottom line. Since opening our doors in 1994, we have earned and maintained a reputation in the industry for our ability to help companies achieve their software-related goals – and for our knowledge of how to do so quickly and effectively.

Our clients span industries, regions and size, including the Fortune 100 and Global 1000. We maintain a North American corporate office in Malvern, P.A. and a European corporate office in the U.K.

For more information, visit www.softwarevalue.com or call 610-644-2856.

Q/P Management Group, Inc.

Massachusetts, USA

Q/P Management Group, Inc. has been a leading provider of software measurement, benchmarking, quality and productivity consulting services for over 25 years. We offer a wide range of software measurement related training, including certified introductory and advanced function point training as well as IFPUG certified SNAP training.

Q/P utilizes the most effective methods and techniques available to assess quality and productivity, implement continuous process improvements and measure results. Q/P's benchmark database is the largest, most accurate source for Function Point (FP) based metrics in the world. The database is comprised of over 20,000 projects and applications from major corporations, commercial developers, and government agencies. The database contains development project and application maintenance statistics for a broad range of tools and techniques utilized by these organizations. Q/P and their clients utilize the data to compare the performance of internal and/or vendor resources against industry benchmarks as a means to identify and measure process improvements. In addition, the data is utilized to determine pricing for commercial software products and outsourcing agreements. The data is also used for estimating software development projects' productivity, cost, schedule, and staffing.

We also offer the Software Measurement, Reporting and Estimating tool, SMRe. SMRe users can generate software development estimates using proven estimating techniques along with historical and/or industry benchmark data. SMRe captures, reports and compares project performance against historical and/or industry benchmark data.

Visit our website, www.QPMG.com for details about our services and product offerings.

High-Efficiency Defect Removal for Software Projects

By Capers Jones, VP, and CTO, Namcook Analytics LLC

Version 5.0 November 26, 2015

Abstract

Software quality depends upon two important variables. The first variable is that of "defect potentials" or the sum total of bugs likely to occur in requirements, architecture, design, code, documents, and "bad fixes" or new bugs in bug repairs. Defect potentials are measured using function points, since "lines of code" cannot deal with requirements and design defects.

The second variable is "defect removal efficiency" (DRE) or the percentage of bugs found and eliminated before release of software to clients. Defect potentials and defect removal efficiency metrics were developed by IBM circa 1973 and are widely used by technology companies. Function point metrics were also invented by IBM during the same time period.

The new SNAP metric is not used in this paper due to the lack of empirical quality data based on SNAP.

Capers Jones, VP, and CTO, Namcook Analytics LLC
Web: www.Namcook.com
Blog: http://Namcookanalytics.com
Email:Capers.Jones3@gmail.com

Introduction

Defect potentials and defect removal efficiency (DRE) are useful quality metrics developed by IBM circa 1973 and widely used by technology companies as well as by banks, insurance companies, and other organizations with large software staffs.

Defect potentials are the sum total of bugs found in requirements, architecture, design, code, and other sources of error. The approximate U.S. average for defect potentials is shown in table 1 using IFPUG function points version 4.3:. Function point metrics were also invented by IBM in the same time period circa 1973.

Function points were invented by A.J. Albrecht and colleagues at IBM White Plains. Defect potential and DRE metrics were

continued on page 6

(continued from page 5)

developed by Michael Fagan, Ron Radice, Capers Jones, and other IBM personnel at IBM Kingston and IBM San Jose to validate the effectiveness of inspections. Function point metrics, defect potential metrics, and DRE metrics were placed in the public domain by IBM.

Function points have become global metrics and responsibility for counting rules passed to the International Function Point Users Group (IFPUG).

Defect potentials and DRE metrics are widely used by technology companies but do not have a formal user group as of 2015. However, these metrics are frequently used in software benchmarks produced by organizations such as the International Software Benchmark Group (ISBSG) and Namcook Analytics LLC. These metrics are also standard outputs from the Software Risk Master (SRM) estimation tool, which was used to produce Table 2 in this report.

Table 1: Average Software Defect Potentials circa 2015 for the United States

• Requirements	0.75 defects per function point
• Architecture	0.15 defects per function point
• Design	1.05 defects per function point
• Code	1.25 defects per function point
• Security code flaws	0.25 security flaws per function point
• Documents	0.45 defects per function point
• Bad fixes	0.60 defects per function point
• Totals	**4.50 defects per function point**

Note that the phrase "bad fix" refers to new bugs accidentally introduced in bug repairs for older bugs. The current U.S. average for bad-fix injections is about 7%; i.e. 7% of all bug repairs contain new bugs. For modules that are high in cyclomatic complexity and for "error prone modules" bad fix injections can top 75%.

Defect potentials are of necessity measured using function point metrics. The older "lines of code" metric cannot show requirements, architecture, and design defects not any other defect outside the code itself. (As of 2015 function points are the most widely used software metric. There are more benchmarks using function point metrics than all other metrics put together.)

The overall U.S. range in defect potentials runs from about 1.50 per function point to more than 6.00 per function point. Factors that influence defect potentials include team skills, development methodologies, CMMI levels, programming languages, and defect prevention techniques such as joint application design (JAD) and quality function deployment (QFD).

Defect removal efficiency (DRE) is also a powerful and useful metric. Every important project should top 99% in DRE, but few do. The current U.S. range in DRE is from below 80% for projects that use no pre-test defect removal and only a few test stages. The highest measured DRE to date is about 99.95%. and this level required pre-test inspections, static analysis, and at least 8 test stages. The current U.S. average in DRE is just over 92% which is only marginal. All projects should top 97% and the best should top 99%.

DRE is measured by keeping track of all bugs found internally during development, and comparing these to customer-reported bugs during the first 90 days of usage. If internal bugs found during development total 95 and customers report 5 bugs, DRE is 95%.

Table 2 shows U.S. ranges of DRE by applications size measured in IFPUG function points:

Table 2: U.S. Software Average DRE Ranges by Application Size

Function Points	Best	Average	Worst
1	99.95%	97.00%	94.00%
10	99.00%	96.50%	92.50%
100	98.50%	95.00%	90.00%
1000	96.50%	94.50%	87.00%
10000	94.00%	89.50%	83.50%
100000	91.00%	86.00%	78.00%
Average	**95.80%**	**92.20%**	**86.20%**

As can be seen, DRE comes down as application size goes up. For that matter defect potentials go up with application size. Large systems above 10,000 function points are very risky due to high defect potentials and low DRE values.

Table 3 shows approximate DRE values for common pre-test and test methods although there are variations for each method and also for the patterns of methods used. Note that table 3 omits architecture bugs due to the small size of the example of only 1000 function points.

Table 3 assumes top-level experts, the quality-strong "team software process" (TSP) methodology, the Java programming language, and CMMI level 5. Therefore, defect potentials are well below current U.S. averages.

To illustrate the principles of optimal defect prevention, pre-test removal, and test defect removal table 3 shows a sequence of pre-test and test stages that will top 99% in defect removal efficiency (DRE). Table 3 illustrates 1,000 function points and about 53,000 Java statements.

Table 3: DRE > 99%	Defects
Requirements defect potential	134
Design defect potential	561
Code defect potential	887
Document defect potential	135
Total Defect Potential	1,717
Per function point	1.72
Per KLOC	32.20

Defect Prevention	Efficiency	Remainder	Bad Fixes	Costs
Joint Application Design (JAD)	27%	1,262	5	$28,052
Quality Function Deployment	30%	888	4	$39,633
Prototype	20%	713	2	$17,045
Models	68%	229	5	$42,684
Subtotal	**86%**	**234**	**15**	**$127,415**

Pre-Test Removal	Efficiency	Remainder	Bad Fixes	Costs
Desk check	27%	171	2	$13,225
Static analysis	55%	78	1	$7,823
Inspections	93%	5	0	$73,791
Subtotal	**98%**	**6**	**3**	**$94,839**

Test Removal	Efficiency	Remainder	Bad Fixes	Costs
Unit	32%	4	0	$22,390
Function	35%	2	0	$39,835
Regression	14%	2	0	$51,578
Component	32%	1	0	$57,704
Performance	14%	1	0	$33,366
System	36%	1	0	$63,747
Acceptance	17%	1	0	$15,225
Subtotal	**87%**	**1**	**0**	**$283,845**

			Costs
PRE-RELEASE COSTS	**1,734**	**3**	**$506,099**
POST-RELEASE REPAIRS (TECHNICAL DEBT)	**1**	**0**	**$658**
MAINTENANCE OVERHEAD			**$46,545**
COST OF QUALITY (COQ)			**$553,302**

Defects delivered	**1**
High severity	**0**
Security flaws	**0**
High severity %	**11.58%**
Delivered Per FP	0.001
High severity per FP	0.000
Security flaws per FP	0.000
Delivered Per KLOC	0.014
High severity per KLOC	0.002
Security flaws per KLOC	0.001
Cumulative Removal Efficiency	**99.96%**

DRE measures can be applied to any combination of pre-test and testing stages. The U.S. norm is to use static analysis before testing and six kinds of testing: unit test, function test, regression test, performance test, system test, and acceptance test. This combination usually results in about 95% DRE.

Critical software for medical devices, avionics packages, weapons systems, telecommunications switching systems, operating systems and other software that controls complex physical devices use full pre-test inspections and static analysis plus, at least, eight kinds of testing. These applications need to top 99% in DRE in order to operate safely.

In order to top 99% in DRE table 2 shows several forms of defect prevention and includes inspections as an important pre-test removal method. Formal inspections have the highest DRE of any known method and over 50 years of empirical data.

Due to inspections, static analysis, and formal testing by certified test personnel, DRE for code defects can top 99.75%. It is harder to top 99% for requirements and design bugs since both resist testing and can only be found via inspections, or by text static analysis.

Summary and Conclusions

The combination of defect potential and defect removal efficiency (DRE) measures provide software engineering and quality personnel with powerful tools for predicting and measuring all forms of defect prevention and all forms of defect removal.

Function point metrics are the best choice for normalizing defect potentials since they can include the defects found in requirements, architecture, design, and other non-code defect origins. The older lines of code metric can only measure code defects which are usually less than 50% of total defects.

continued on page 8

(continued from page 7)

REFERENCES AND READINGS ON SOFTWARE QUALITY

Beck, Kent; Test-Driven Development; Addison Wesley, Boston, MA; 2002; ISBN 10: 0321146530; 240 pages.

Black, Rex; Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing; Wiley; 2009; ISBN-10 0470404159; 672 pages.

Cohen, Lou; Quality Function Deployment – How to Make QFD Work for You; Prentice Hall, Upper Saddle River, NJ; 1995; ISBN 10: 0201633302; 368 pages.

Everett, Gerald D. And McLeod, Raymond; Software Testing; John Wiley & Sons, Hoboken, NJ; 2007; ISBN 978-0-471-79371-7; 261 pages.

Gack, Gary; Managing the Black Hole: The Executives Guide to Software Project Risk; Business Expert Publishing, Thomson, GA; 2010; ISBN10: 1-935602-01-9.

Gilb, Tom and Graham, Dorothy; Software Inspections; Addison Wesley, Reading, MA; 1993; ISBN 10: 0201631814.

Jones, Capers and Bonsignour, Olivier; The Economics of Software Quality;

Addison Wesley, Boston, MA; 2011; ISBN 978-0-13-258220-9; 587 pages.

Jones, Capers; Software Engineering Best Practices; McGraw Hill, New York; 2010; ISBN 978-0-07-162161-8;660 pages.

Jones, Capers; Applied Software Measurement; McGraw Hill, 3rd edition 2008; ISBN 978=0-07-150244-3; 662 pages.

Jones, Capers; Estimating Software Costs; 2nd edition; McGraw Hill, New York; 2007; 700 pages..

Kan, Stephen H.; Metrics and Models in Software Quality Engineering, 2nd edition; Addison Wesley Longman, Boston, MA; ISBN 0-201-72915-6; 2003; 528 pages.

Nandyal; Raghav; Making Sense of Software Quality Assurance; Tata McGraw Hill Publishing, New Delhi, India; 2007; ISBN 0-07-063378-9; 350 pages.

Radice, Ronald A.; High Quality Low Cost Software Inspections; Paradoxicon Publishing Andover, MA; ISBN 0-9645913-1-6; 2002; 479 pages.

Wiegers, Karl E.; Peer Reviews in Software – A Practical Guide; Addison Wesley Longman, Boston, MA; ISBN 0-201-73485-0; 2002; 232 pages.

Thinking Outside the Box Using FPA on Emerging Technologies

By Ricardo Gaspar, CFPS

Brazilian government software development contracts must use metrics - in most cases, function points - to derive price, productivity and quality criteria.

The Brazilian Development Bank (BNDES) has contracts to develop software using emerging technologies, such as Portlets on web sites, Business Process implementations and an Enterprise Service Bus layer. As a government company, those contracts follow the use-of-metrics determination (function points). But how these technologies should be considered in function point counting? Since counting function points should be a technology independent process, how to measure the work and establish a price on contracts for software built on these technologies?

Definitions

In order to answer these questions, we need to review some definitions:

Portlets

On the Web, a portlet is a component of a portal Web site that provides access to some specific information source or application, such as news updates, technical support, or an e-mail program among many other possibilities.

BPMS

A Business Process Management Suite (BPMS) is a tool for designing, implementing and improving an activity or set of activities that will accomplish a specific organizational goal.

Enterprise Service Bus

An Enterprise Service Bus (ESB) is a software architecture for middleware that provides fundamental services for more complex architectures. For example, an ESB incorporates the features required to implement a service-oriented architecture (SOA).

How to count function points in this scenario?

The answer is to think outside the box. CPM should always be followed, but local guidelines and extensions are required to use function point analysis on emerging technologies. A good hint is to search the IFPUG white papers, which provides a theoretical basis for creating the local extensions and guidelines.

On BNDES, there are specific guidelines for counting Portlets, BPMS implementations and the ESB layer based on a user perspective. IFPUG's white paper "Function Points

& Counting Middleware Software Applications" was the reference for the counting guidelines to size BPMS and ESB, while CPM 4.3.1 was used to establish specific guidelines for correct interpretation on sizing Portlets.

None of the guidelines conflicts with CPM, they just intend to present a new user-based perspective on scenarios not detailed by CPM and present hints on how to apply Function Point Analysis (FPA) to establish the function point size of these technologies.

Thinking Outside the Box

1) BPMS and ESB

According to "Function Points & Counting Middleware Software Applications" paper, "the term middleware is used to describe products (software) that serve as the glue between two applications". A middleware is defined as an application itself and other applications are its users.

Based on this definition, the service layer on an enterprise service bus can be considered middleware software, connecting two sides of different applications and passing data between them. The same interpretation can be used to BPMS implementations, as they provide workflow functionality to integrate different applications and processes.

Based on this perspective, the following guidelines were created:

• Determine the boundary

Applications that interface with the middleware are the users and since the purpose is to size the enterprise service bus or the BPMS implementation, the boundary is separated from the application and limited to the middleware-provided functionality.

• Determine the data functions

Internal logical files and external interface files must be counted according to CPM and consider the middleware boundary.

On BNDES implementation of ESB, data is passed between applications and validated on the service layer, but not stored on it. That's why in most cases, there are no ILFs to be counted. However, there may be a log file if it applies to CPM rules.

Concerning BPMS, based on the user perspective, there are two ILFs per workflow implementation: "process instance" and "process configuration". "Process instance" stores the current state of the process on the workflow, while "process configuration" stores the information that final user inputs into the system to configure the workflow.

Important: the number of BPMS implementations of workflows must be considered based on the user's view.

• Determine the transactional functions

External queries, external outputs and external inputs must be counted according to CPM and consider the middleware boundary.

On BNDES implementation of ESB, data is passed between applications and validated on the service layer, but not stored on it. That's why in most cases, there are no EIs to be counted. If there is an ILF "log", the respective EI must be counted.

Concerning BPMS, based on the user perspective, there are two EIs and one EQ per workflow implementation: one EI to maintain the "process instance" ILF, one EI to maintain the "process configuration" ILF and one EQ to pass process information to the systems that interface with BPMS.

• General guidelines

  o Middleware's functional size must not be added to the application baseline functional size.

  o As a result of boundary positioning, middleware (ESB or BPM) functionalities must be counted only once, even if different applications use them.

2) Portlets

Portlets can usually be sized using CPM 4.3.1. However, as they provide access to some specific information source or application functionality, some aspects must be considered:

continued on page 10

(continued from page 9)

Has the portlet exposed functionality already been counted in other application? Was the portlet generated automatically by software or was there functional customization?

The following guidelines were created based on these considerations:

- Determine the boundary

Based on the user's view, portlets boundaries are positioned inside the same boundary of the application that had its functionality exposed. There must not be specific boundaries established for portlets.

- Determine the transactional and data functions

Internal logical files and external interface files must be counted according to CPM, considering the positioned boundary.

Concerning transactional functions, if the functionality exposed by the portlet is part of another system, it should not be counted, unless it has suffered functional customization. Examples of functional customization may include additional (or less) data element types and different processing logic.

- General guidelines
    - o Automatic generated content must not be measured. Portlet automatically generated by software is considered part of the "generator" boundary and should not be sized on the web portal boundary.

An example of the use of the guidelines is shown below.

Study Case

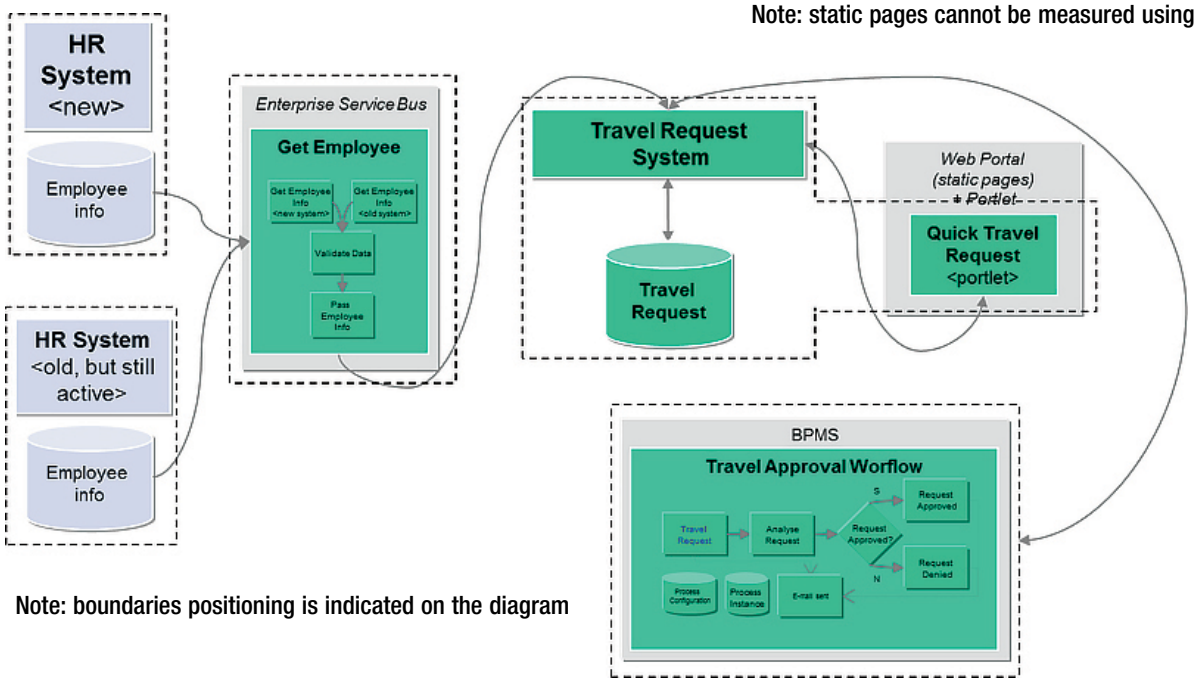
In this example, we need to develop a new travel request system to a company using all the elements of this architecture:

- Employees must request a travel to his company through a new transactional system. Travel request can be updated and deleted before sending for approval.
- There are two HR systems that store employee's information that are necessary to request a travel. Depending on when the employee was admitted, his information may be stored in the new system or in the old one. The service layer (ESB) decides where to get the information and validates it.
- Travel request is approved in a workflow implementation on BPMS.
- A portlet is available on a web portal to make a quick travel request for simpler requests.

Based on the guidelines, the function point counting would be:

Data or transactional function	Type	Details
Travel Request	ILF	Transactional system
Request travel	EI	Transactional system
Update travel request	EI	Transactional system
Delete travel request	EI	Transactional system
Send request for approval	EQ	Transactional system
Employees	EIF	Transactional system
Get employee info	EO	ESB
Travel Approval Process instance	ILF	BPM
Travel Approval Process configuration	ILF	BPM
Update state of "approval process"	EI	BPM
Update configuration of "approval process"	EI	BPM
Get "approval process" state	EQ	BPM
Quick travel request	EI	Portlet

Note: static pages cannot be measured using function points



Note: boundaries positioning is indicated on the diagram

Conclusion

There is effort on developing Portlets on web sites, BPMS implementations and an Enterprise Service Bus layer, as in other emerging technologies. On contractual relations, the work done must be paid and it can be measured using function points.

If you need to measure emerging technologies, you need to think outside the box to establish specific guidelines, without disrespecting what is stated in CPM.

REFERENCES

[1] Counting Practices Manual, International Function Point Users' Group (IFPUG). 4.3.1.

[2] Definition of Portlet available on the web site: http://searchsoa.techtarget.com/definition/portlet

[3] Definition of Business Process Management Suite (BPMS) available on the web site: (http://searchcio.techtarget.com/definition/Business-process-management-suite-BPMS)

[4] Definition of Enterprise Service Bus (ESB) available on the web site: (http://searchsoa.techtarget.com/definition/enterprise-service-bus)

[5] White paper "Function Points & Counting Middleware Software Applications", International Function Point Users' Group (IFPUG

What's the Story (points) with Function Points

By David Herron

Story points or function points or both? That question is coming up more and more as organizations become increasingly invested in agile. Some common questions and comments we hear include:

Can I use function points on an agile project?

Story points are much easier and faster than function points.

Is there a relationship between story points and function points?

Before we get to the answers, some ground work is necessary to put these two measures into perspective.

Story Points

According to https://tcagley.wordpress.com, story points are a relative measure based on the team's perception of the size of the work. The determination of size is based on a level of understanding, how complex and how much work is required compared to other units of work. Story points are expressed according to a numerical range, which is usually constrained to a limited set of numbers such as an adaptation of a Fibonacci sequence (e.g. 1, 2, 3, 5, 8 etc.).

Story points are a relative measure used by agile teams typically during a sprint session. Each story is assigned a story point value based on everyone's best understanding as to the "level of difficulty" of that particular story. Of course, "level of difficulty" can include different things such as complexity, size, duration, effort and so on. Regardless of the scale being used, in a process called planning poker, the values assigned are assessed independently by each individual, compared by the team and then discussed to reach a consensus. There is no consistent definition of what the values represent other than to use it as a comparative value of one story being larger/harder or smaller/easier than another within the team. Over a number of iterations (sprints) an agile team can develop a consistent velocity (number of story points delivered per sprint) which can serve to estimate future amounts of work/effort in future sprints. Of course, even if one team is achieving exactly the same volume/complexity of work as another team, their story points will not necessarily be the same.

Function Points

"Function points measures software by quantifying the functionality requested by and provided to the customer, based primarily on logical design"; as defined by the International Function Point Users Group (www.ifpug.org). Function points measure "software size" or, more precisely, the size of the requirements/design specified to which the resulting software provides a "no more, no less" solution. The size of a defined business requirement is a necessary piece of information if you want to estimate how long it will take and how much effort it will take to develop that piece of software. Unlike story points, function points are a defined, reproducible unit of measure. They can be measure consistently regardless of who is measuring them. Function points can be used on both agile and non-agile projects. For example, agile user stories, for the most part, describe the features and functions requested by the product owner.

The function point methodology calls for the identification of 5 key elements including inputs, outputs, inquiries, interfaces and internal stores of data. Naturally there needs to be some

continued on page 12

(continued from page 11)

description of these elements; e.g., requirements documentation or stories, in order for a function point sizing to be accomplished. Once a function point size is determined it can be used to estimate level of effort or on the backend, the size information can be used to calculate productivity (fp/effort hours) and quality (defects/fp) levels of performance.

Some Answers…

Can I use Function Points on an agile project?

Yes, function points can be used on an agile project. In fact, both story points and function points can be used on agile projects and can serve to effectively manage the project and measure performance.

We already know that story points are used to size the user stories for a given sprint/iteration. Stories can also be sized using function points. However, you don't need to use function point size to estimate how long a collection of stories in a sprint are going to take because you have already set up a 2, 3, or 4 week cadence for your sprints.

Function Points are most useful and frequently used at the beginning of an agile project and upon delivery of a release or some significant delivery of functionality. In the beginning of an agile project, you may use function points to size the entire backlog and use that size information along with additional historical data points to estimate a total project cost and a predicted delivery date. At the backend of the project, you may capture total function points delivered to look at performance levels and compare agile project performance levels to performance levels of other methodologies currently in use.

Story Points are much easier and faster than Function Points

This is a true statement; story points are quicker and easier than function points. The question really becomes, which method is more appropriate for the task at hand. Sitting down with the agile team and assigning story points to selected stories for a sprint backlog is an excellent exercise in approximating the complexity and required effort of selected stories. This is a collaborative approach that involves the team and provides a group understanding of each work element (story) and what may be involved. Even if story points were not assigned, the discussion alone would be of significant value in driving team efficiency.

Function points require a more detailed examination of the information (stories) available and achieving reproducible counts requires expertise and practice. There are specific guidelines to be applied and calculations to be made. It may be unrealistic to expect every team member of an agile team to have this skill set. As a result, the use of function points throughout an organization is usually performed by a central specialist team thus allowing for comparisons among the various agile teams and portfolios. Function points are also a size measure that serves both the developer and the end user. For the developer, they are used to manage the project outcomes. For the end user (product owner) function points can be a useful vehicle for setting expectations with regard to identifying (and agreeing) what features and functions are being developed and deployed. However, the direct involvement of the agile team members in sizing the tasks they are going to work on has motivational benefits over the seemingly imposed sizing of the Central FP counting team.

Easier and faster are nice, but that is not the issue. The issue really is all about which metric or set of metrics will provide you with the information you need to best manage the software deliverable, to make decisions and to manage expectations.

Of course, the real issue with the speed and ease of story points is that they are hard to scale across many agile teams. For the agile teams themselves this is not an issue but for the organization which needs to build product road maps, annual budgets, resource plans and so on, the loss of coherence is a significant one.

Is there a relationship between story points and function points?

The narrative below references the following example…

Iteration	# of stories	Story Points	Function Points	Complexity
1	10	50	100	Simple
2	5	50	25	Complex

Iteration 1 – the team completed 10 stories (in a two-week sprint) that were assigned a total of 50 story points. The function point size for those 10 stories was 100. The stories were focused on simple transaction I/O processing.

Iteration 2 – the team completed 5 stories in their second two-week sprint. The stories were assigned 55 story points in total. The function point size for those 5 stories was 25.

Question: Assuming the team has achieved a fairly consistent velocity (50) why isn't there a correlation between SPs and FPs?

Observations:

Story points are often assigned with some consideration of required level of effort. In the first iteration, the stories involved fairly simple processing and therefore were assigned an average of 5 story points each. In the second iteration, the stories represented more complex processing and were assigned an average of 10 story points.

Function Point analysis does not consider level of effort. It is accounting for the features and functions being delivered. The stories in iteration 1 were about processing inputs and outputs and accounted for a high number of function

points. In the second iteration, the stories required a greater degree of processing logic, but the features and functions being delivered were fewer.

Story Points are a relative measure whereas function points are a well-defined consistent method of sizing.

Does this mean that function points cannot be used to estimate at a sprint level? Sprints are time boxed, usually as two-week iterations. The desired state is to achieve a steady flow of work from sprint to sprint (velocity). For agile teams, this is adequately measured using story points. Function points are more appropriately applied to measure the overall project outcome. This can be done upon delivery of a release and/or function points can be applied when the product backlog is first developed as a means to estimate the total level of effort that may be required across all sprints.

Summary

Story Points vs Function Points; so do we settle on one or the other or both? The answer is both. Both these measures are useful and serve the intended purpose to more effectively manage a software deliverable.

Function Points are good for measuring the overall product deliverable at the beginning and at the end. The FP size information at the beginning of a project can be used to estimate overall schedules and costs. And the size information upon delivery can be used to measure performance.

Story Points are an effective method for managing the flow of work in an agile project. They too serve a purpose of estimating the amount of work that can be accomplished by the team in a defined period of time (sprint/iteration).

Clearly, sometimes the best use of these two methods overlaps and so it is important to make strategic decisions about when and how they will be used rather than local, tactical decisions.

Consistency is King: Counting FP for Agile / Iterative Software Development Projects

By Carol Dekkers, CFPS (Fellow), PMP, AEC, CSM

Abstract

One of the biggest benefits of Function Point Analysis (FPA) lies in its ability to objectively and consistently measure software size without regard to the underlying technology or project methodology. As such, function points can be used to size the software developed using any development approach from waterfall to spiral to agile to hybrid. This independence suggests that function points are ideal to compare (productivity and quality across) different types of projects, but one must use caution to ensure that measurement tells the truth – that measurement is based on consistent rules across all types of projects. While the IFPUG counting rules are consistent and published (currently IFPUG 4.3.1 is the standard,) inconsistencies come to light in the *application of FP rules* in agile/iterative development versus waterfall or other development. These inconsistencies, if not addressed, can create misleading results and confusion for cost, productivity, and schedule evaluations that span multiple software delivery methods.

This article identifies the root cause of inconsistencies and examines IFPUG definitions (including "project," "elementary process," "consistent state," and "enhancement") as they apply to modern project approaches where terms like "sprint," "iteration," "release," or "story map" prevail. The goal is to marry IFPUG definitions with equivalent concepts in agile/iterative processes and create a basis for consistent comparisons across all types of projects.

Introduction

When estimating the cost, effort and duration of software development projects (labeled by IFPUG as *development projects* for the first release of a software product and *enhancement projects* for its subsequent adaptive maintenance and enhancement) the functional size of the software is a pre-requisite input. Similar to using a floor plan (sized in square feet) as the basis for estimating a building construction project, the functional size of software (to be delivered) is a solid basis for estimating a software development project. FP are a reliable 'common denominator' for comparing project productivity (FP/effort), duration delivery (FP/elapsed time), maintainability (hours/FP), product quality (defect density) and other important aspects of software delivery.

With today's IT landscape dotted with agile, iterative, spiral, waterfall and combination approaches to development, businesses are searching for the ultimate approach to delivering the right software (functionality- and quality-wise) at the lowest unit cost for the least amount of effort. But for estimation to succeed in this climate, we'll need consistent FP definitions across all methods of software delivery.

Agile Is Here to Stay

Gone are the days when agile/iterative development methods were considered "rogue" and without structure; today, agile

continued on page 14

(continued from page 13)

methods are held in high esteem, even in conservative software development shops where waterfall still prevails. Indeed, the penetration of agile in the IT marketplace has had numerous positive impacts, including:

- Users are more receptive to participating on projects and better understand the impact that their non-involvement can cause;
- Business stakeholders are more engaged; and
- Developers can be more responsive to changing business requirements.

Equipped with both agile/iterative and waterfall methods, business leaders want to know the optimal combination of talent, tools, techniques, cost, and schedule to deliver good-enough-quality software for a reasonable investment of time and money. Finding that "sweet spot" relies on consistently measuring the same elements in the same ways across various delivery methods. Consistency in measurement depends on consistent definitions and in applying measurement methods such as function point analysis.

The first step is to align the IFPUG definitions and then examine how to apply them consistently to all types of projects.

IFPUG Definitions

The IFPUG FP methodology (IFPUG 4.3.1) provides guidance to count FP on projects by identifying delivered 'elementary processes that leave the business in a consistent state.' Agile/iterative techniques deliver software incrementally through iterations or sprints within a project. Defining what constitutes a "project" and the delivery of an "elementary process" in agile/iterative is the KEY element for consistent function point counting across development approaches.

Since FP counting of software *development* is meant to measure the size of the functional user requirements delivered via a project *(development or enhancement)*, the methodology used to implement that functionality (be it agile/iterative, spiral waterfall, or any other development method) should have no effect on the size of the delivered software product. The application FP (also called the baseline or installed application FP size) is the same regardless of the delivery method used and can be measured consistently at the completion of any type of software delivery. Application FP counts are of secondary concern for this paper; the primary concern is defining what constitutes a "project" (either development or enhancement) in agile/iterative development.

Terminology Presents Challenges

Before we get into the issues and challenges of counting FP in an agile environment, let's add a bit of strictness and consistency to a few of our terms:

- **Release:** A release is the distribution of the final version of an application. A software release may be either public or private and generally constitutes the initial generation of a new or upgraded application. A release is preceded by the distribution of alpha and then beta versions of the software. In agile software development, *a release is a deployable software package that is the culmination of several iterations.* (source: http://searchsoftwarequality.techtarget.com/definition/release)
- **Project:** *A collection of work tasks with a time frame and a work product to be delivered* (IFPUG 4.3.1 glossary). According to the Project Management Institute (PMI.org), a project is a *temporary endeavor undertaken to create a unique product or service.*
- **Iteration:** In agile software development, an iteration is a *single development cycle, usually measured as one week or two weeks.* (source: http://whatis.techtarget.com/search/query?q=iteration)

(Side note: Some proponents of agile insist that all iterations be the same length and that the particular length of iterations (anywhere from 2 to 6 weeks) is of less importance. For our purposes in this paper, the key element is that an iteration represents a single development cycle.)

Sprint (software development): In product development, a sprint is a *set period of time during which specific work has to be completed and made ready.* (source: http://whatis.techtarget.com/search/query?q=sprint)

For **waterfall development,** it is fairly easy to identify and count FP for discrete development and enhancement projects. "Release" and "project" are often used synonymously to refer to the scope of a self-contained software delivery.

For **agile development,** a "project" is not so easily identifiable. "Sprint" and "iteration" are used more often than "release," and those terms are based on elapsed calendar time or work effort rather than functionality. "User stories" (or "use cases") are used to describe functionality and are useful for identifying functional user requirements, but there is no requirement that they constitute an elementary process or that they leave the business in a consistent state – both of which are required for FP. The notion of using "story points" (a sizing approach intended to quantify the relative size of a user story) as equivalent to FP (as suggested by a few agile advocates) is not feasible for the following reasons:

- Story points are not convertible to FP (there is no conversion factor);
- Story points are not standardized (FP are standardized through IFPUG and ISO); and
- Story points capture user story size differently (and based on different concepts) than FP.

When functionality is delivered in discrete and well-defined construction projects, as is intended with **waterfall-style deliveries,** counting FP is easy and based on a single set of functional user requirements. Even when a subset of the overall features is delivered in one release and then enhanced in a later release, the discrete "chunks" of new/enhanced functionality make counting FP a straight-forward process using IFPUG methodology.

With traditional waterfall delivery, the terms "developed" and "delivered" are almost always used interchangeably. But in **agile/iterative development,** there is a difference between "developed software" (which is not yet ready for mass deployment) and "delivered software" (ready for full deployment).

Therefore, with agile/iterative forms of software delivery, counting the "delivered" functionality is not so easy. IFPUG (and other ISO conformant) functional size measurement techniques are counted based on complete elementary processes or functions that leave the business in a consistent state, not on parts thereof. A "function point" count consists of delivered (or anticipated to be delivered) functions that are whole business processes. Partially delivered functions that are incomplete and cannot support the business without further work would not typically be counted as "function points delivered." For example, a business process such as create hotel reservation would not be an elementary process until a reservation is made and stored. If the first part of the reservation process was delivered in one sprint (such as checking the availability of a hotel for given dates) and the latter part was delivered subsequently (enter customer information and book reservation), FPs would be count on the elementary process (both parts.)

Why Does All This Matter?

At this point, you may be asking why we don't just use the word "release" in place of the word "project" and be done with it.

Or, couldn't we simply count up the delivered function points at the end of a "release" no matter how the software is developed, and then compare the productivity across releases to perform our estimates?

Actually, yes. This is absolutely the right way to go, but only if our definition of "release" can remain consistent across our various forms of delivery. For starters, we'll have to determine the number of agile iterations or sprints that constitute a "release." Let's consider the following situations:

- When a single software "product" (i.e., the result is a working piece of software) is delivered via two or more distinct releases, each of which is completed and implemented into production (i.e., fully-functioning software), the software *application* in place at the end of the two releases is exactly the same size as it would have been if delivered all at once. (Consider the analogy of a floor plan that is built in stages versus all at once – the resultant square foot size is the same.) Each release is discrete and self-contained, and the sum of the FP of the two releases likely will exceed the installed application base (because some of the functionality completed in release 1 may be enhanced through adaptive maintenance FPin release 2, yet may not increase the application size (also called "installed base" size or baseline). Think of how a house can be delivered in a first construction and then renovated in a second – the square foot size of the two constructions added together may exceed the overall size of the house.
- However, when the software is built iteratively over the course of a year in two-week sprints, there is a lack of discrete delivery. (Think of building a house bits at a time and slowly developing the underlying floor plan.) Where is the "elementary process" for FP counting? Likely, the "complete" functions were delivered through multiple user stories or use cases spanning a number of sprints or iterations. Therefore, the challenge to counting function points in agile/iterative lies in the question of when and where a business process or function leaves the business in a consistent state.

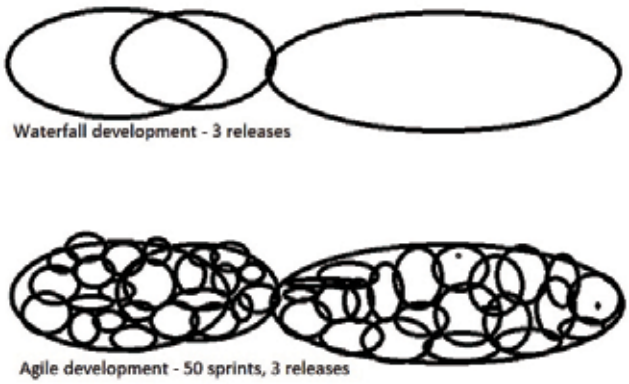


Diagram 1 Waterfall releases vs agile releases

Some would suggest that each sprint or iteration should count as a discretely countable functionality in FP; however, this approach would contradict FP definitions and concepts including:

- Elementary process
- Self-contained
- Enhancement (defined as the adaptive maintenance of a delivered software product)
- Functional user requirements (which relies on elementary processes)
- Development *project*
- Enhancement *project*

continued on page 16

(continued from page 15)

- Counting scope (a set of Functional User Requirements)
- Base Functional Requirement (BFC) – elementary unit of Functional User Requirements
- Consistent state
- Why would counting "sprints" violate these definitions? The answer is that a sprint, by definition, is based on elapsed time or work effort – not on functionality. Certainly user stories (and use cases) describe user functionality, but there is no requirement that they describe complete or self-contained functionality. Consider that two user stories for an airline reservation system might be written as:

a. Choose the flights on which you want a seat

b. Pay for the reservation

Clearly these are two different steps in the sequence of steps needed to complete the airline reservation. But each one is NOT its own separate elementary processes they're part of a single elementary process where both steps are needed to leave the business in a consistent state.

Further examples:

Release #	Release (project) FPs	Application FPs
1	300	300
2	250 (200 new + 50 chg)	500

FPs in Agile/Iterative Development

When we talk about implementing user stories or use cases, the assumption may be that each user story or use case equals at least one standalone and self-contained function. This is seldom the case. While the scope grows and morphs with each iteration or sprint, the requirements are often progressively elaborated. This means that one "functional user requirement" may span multiple user stories or use cases – especially if it is a complex one. Thus, the application may not satisfy the full user requirement for a process or transaction until after a set of sprints is implemented.

A full agile software development implemented in a series of two- to six-week sprints will deliver functionality in a piece by piece fashion. It may not be easy to determine/predict when the functionality will actually be delivered, especially once the IFPUG definitions for "elementary process" and "leaving the business in a consistent state" are taken into account.

Using the home building analogy, agile development is similar to pouring the foundation and building rooms a bit at a time, as the overall floor plan eventually comes into being. When a home is constructed in this manner, it is not ready to be occupied until the rooms are finished and a roof covers the structure. In agile development, functionality is typically delivered partially – in sprints – and it isn't until several sprints are delivered that the business can begin to actually use the software. Yet there is a tendency to assume that sprints and iterations are akin to a new development project for the first sprint and enhancement projects for each sprint thereafter.

The challenge to counting FP on agile projects lies in determining which functional user requirements have been satisfied (and *when* they are satisfied) by the software delivery.

For instance, if a function is "delivered" in a sprint (i.e., we count FP for the initial sprint, assuming that the user story completely describes an elementary, self-contained, and complete business function), and subsequently enhanced in a second sprint, was the original function:

a. Incomplete (i.e., the elementary process was NOT fully delivered in the first sprint) – and we shouldn't have counted/taken credit for FP in the first sprint?

b. Complete at the time of the first sprint but now enhanced due to changing requirement – and we should count delivered FP for sprint #1 and count the entire transaction's FP a second time for sprint #2?

c. Flawed in the first sprint – therefore, the FP counted in sprint #1 should not be recounted in sprint #2 (because sprint #2 was only corrective maintenance)

d. Some other variation?

The value and beauty of FP are that they provide a methodology- and technology-independent assessment of software size based on the functional user requirements, which (at the end of both agile AND waterfall development) are the same. The actual size of the installed application baseline (FP installed) is – or *should* be – the same, regardless of HOW the software is developed.

When does the delineation of FP across sprints become an issue? (Or, why is it important to count delivered FP in a consistent manner regardless of development methodology between agile and waterfall projects?)

There are two significant situations where the FP "delivery" is critical to businesses:

1. **Productivity assessment.** Businesses want to compare the cost per FP or effort per FP between agile and waterfall projects, but doing so requires a consistent baseline. If we count FP for each sprint the same way as we do for an entire project, the total project FP delivered in agile (the sum of FP across all sprints) may be 10 or more times the total project FP delivered using waterfall (the sum of FP across several releases) thereby invalidating productivity comparisons;
2. **Outsourcing.** When businesses commit to paying for software as it is "delivered", it makes no sense that the business should pay over and over for partial delivery of functionality just because it is delivered using an agile approach. From the client perspective, the overall delivered software (base) is the same size,

Therein lies the dilemma and the challenge in using function points on agile projects – it is problematic to credit agile projects

that deliver same completed functionality (i.e., complete elementary processes leaving the business in a consistent state) with having produced MORE functionality than waterfall projects!

To recap, let's look at one example using the two different methods:

Waterfall software delivery: The business needs a new customer service application where the final installed software will equal 1000 FP. Through negotiation and agreement, three phases/projects are outlined, each of which delivers working software in production.

1. Phase 1: New development project = 300 FP. (Installed baseline at the end of the project = 300 FP.)
2. Phase 2: New functionality of 200 FP and enhancement of 50 that were already delivered in phase 1. Project count = 250 FP. (New installed baseline at the end of the project is now 500 FP.)
3. Phase 3: New functionality of 500 FP and enhancement of 50 that were already in place. Project count = 550 FP. (Installed baseline at the end of the project is now 1000 FP.)

Agile development: The business needs a new customer service application delivered using an agile approach. User stories are iteratively documented and a series of 2-week sprints is agreed upon to allow developers and the business to discover the requirements and define them as they go. Twenty-five different sprints are worked on over a year period, and at the end of the "project(s)" the installed baseline software is 1000 FP.

1. Sprint #1 outlines the need for users to sign in and validate their password. (It is not yet certain where the data will reside. We cannot yet count a datastore definitively but it is envisaged that it will be maintained in either an ILF or EIF in a future user story/user case.) Sprint 1 also delivers the first of several screens needed to set up a new customer.

– Estimated FP count = 1 Low Complexity Query (for user validation) + 1 Average complexity datastore (for customer) + 1 Average Input process (create customer) = 16 FPs. (Baseline = 16 FPs.)

2. Sprint #2 adds a second screen of data for customer creation and identifies the need to allow changes to and deletion of customer records. Customer details (all of the data added across both screens) can be displayed. What should be counted in Sprint #2?

a. The new functionality introduced: Change customer = 1 Average complexity Input; Display customer = Average complexity Query; Delete customer = 1 Low complexity Input EI;

b. The datastore called Customer - it was already counted in the first sprint (and it is the same complexity.). Should it be counted again in sprint #2? It seems nonsensical to do so.

c. The add customer function – it was delivered partially in the first sprint because there wasn't enough time to deliver it fully. It was incomplete (i.e., did not leave the business in a consistent state) in the first iteration – the question is whether the FPs should be counted in sprint #1, in sprint #2, divided between sprints (1/2 and ½ perhaps) or as the entire number of FPs in both sprints (i.e., appearing as double the FPs.)

Guidance on FPs counting For Agile

The following list of recommendations is provided to increase consistency across the various forms of software delivery:

1. Identify the user stories and use cases that contribute to a single elementary process, group them together, and count the FPs for the elementary process (and document what contributed to the function);
2. Count an ILF only when its maintenance is introduced and consider future DETs and RETs it will include (i.e., count a Customer ILF only when the first transactional function to maintain it is delivered, and count its complexity based on all DET and RET envisaged in that release);
3. Count functionality at a release level according to #1.
4. Count as development project FP all functionality for the first release as long as working software is implemented (i.e., users can input data) and elementary processes are complete;
5. Count as enhancement project FP all functionality for subsequent releases as long as working software is implemented and adaptive maintenance is performed on each release;
6. If data or transactions describe code data, do not count (not as ILF, EIF, or any associated maintenance or query/drop down functions for such data). This needs to be spelled out because this is easy to overlook when counting from use cases or user stories.
7. Document your assumptions used in the FP count(s).

Comparative and consistent FP counts across various development "projects" can be done through consistent terminology, and the application of FP rules. Being careful not to size "bits" of functionality partially delivered, and instead grouping use cases and user stories into elementary processes according to IFPUG 4.3.1 will go a long way to creating consistent FP counts.

Carol Dekkers is president of Quality Plus Technologies and holds credentials as a Certified Function Point Specialist (CFPS – Fellow), a PMP, and Certified Scrum Master (CSM) and Agile Expert Certified (AEC.) She currently serves on the IFPUG Board of Directors as the Director of Communications and Marketing and works as a consultant and instructor in both the public and private sectors, and is a freelance consultant for QSM, Inc. She can be reached by email at dekkers@qualityplustech.com

Boundaries and Partitions in SOA Architectures

Abstract

Correctly placing boundaries is a key activity in software measurement, both for functional and non-functional assessment processes. Professionals involved in software measurement are often facing Service Oriented Architecture (SOA) software architectures, where programmers apply service-oriented design even at fine grain level, in order to meet Functional User Requirements (FURs) AND build "nice", reusable and flexible functions. FPA sizing alone has difficulties in correctly measuring the developing effort consequence of a pervasive use of SOA approach, and SOA architecture is often cause for controversy in a contract: the supplier would count all the services, the SW customer none.

Combined use of FPA and SNAP, through correct and agreed placing of boundaries and partitions, can lead to more appropriate measurements and less controversial contracts. Starting from some examples in literature [1], we will add some further considerations and examples using SNAP.

Context

Two examples of boundaries in SOA architectures are presented:

1. The FUR explicitly requests a set of services to handle the data. There is the need to offer the services to other applications as a separate set of functionalities.
2. The FUR explicitly requests only some services to handle the data. There is the need to offer some services to other applications as a separate set of functionalities, not all the services are public.

But, what if the (set of) services are to be offered to functional components or modules within the application boundary? In this case, a SOA approach is not merely a "technical" consideration, and even if data do not cross the external "membrane" the user has a clear perception of the inner service, and a possible business advantage from a single point of maintenance/evolution for the services.

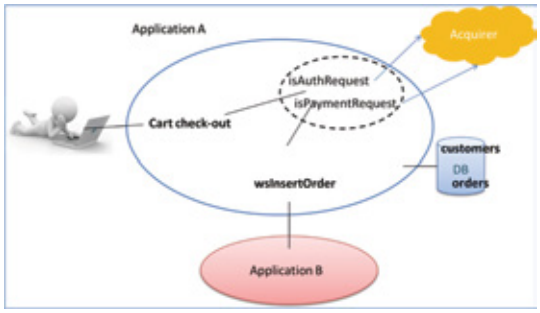
Analysis and Solution

Example 1: internal services

Consider an e-commerce application providing purchase order issuing to web users through its own web interface and to third party sites through a web interface. For the purpose of this discussion, we will focus on the functionalities supporting credit card payment.

The user requirement is that the application has to provide an interactive cart check-out process through web GUI, and a service to submit orders from other applications (in this case, collecting necessary information is external application's responsibility). As for credit card payments, interface towards the acquirer should be easily maintainable.

From the functional point of view, we can identify two elementary processes:



1. Cart check-out: this process provides a multi-step web interface to check cart content, collect customer identity, shipping and invoice addresses, and payment information, then issues an authorization request to the credit card acquirer; in case of positive response, performs a final check with web customer and then issues the request for payment to the acquirer; in case of positive response the order enters the provisioning workflow.
2. wsInsertOrder: this web-service exposes a call interface to submit customer data and payment information; once called the payment request is issued, and then returns the result to the caller; in case of a positive outcome, the order enters the provisioning workflow.

The two elementary processes are distinct (at least for internal logic); they are External Inputs, and we can envisage a high complexity as they access/maintain at least 2 FTRs (customer and order) and deal with more than 15 DETs (customer data, address data, payment data, and cart content). Without considering logical files, through Function Point Analysis we can measure this software impact as 12 FPs, which might be e very strict measure with respect to the effort actually needed to build such functionality (and to the value added to customer's application portfolio).

Now we can consider the Non-Functional User Requirement (NFR) concerning maintainability of the interface with the acquirer. A possible solution might be to follow a SOA approach for technical architecture, implementing a common set of internal interface services to access acquirer's own services, and we can identify at least two internal services:

1. isAuthorizationRequest: this service receives an input payment data (card issuer, card number, card expiration, name on card and card verification value – CVV), issues an authorization request to the acquirer, waits for response for a configurable timeout, and then

returns some data to the caller: result (OK, KO, TIMEOUT), optional KO reason, authorization code)

2. isPaymentRequest: this service receives an input payment data (card issuer, card number, card expiration, name on card and card verification value – CVV), issues a payment request to the acquirer, waits for a response for a configurable timeout, and then returns some data to the caller: result (OK, KO, TIMEOUT), optional KO reason, payment code)

The cart check-out process, in its internal logic, will call both isAuthorizationRequest and, after customer confirmation, isPaymentRequest. The wsInsertOrder service will just call isPaymentRequest.

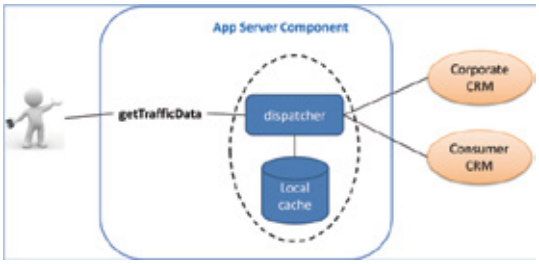
From the functional point of view, the internal services cannot be considered autonomous transactions, and, therefore, cannot be counted as Elementary Processes. But they can be considered as a "set of software functions within an application boundary that share homogenous criteria and values", in other words a SNAP partition [2].

Therefore, applying SNAP, we can consider category 1.4 – Internal Data Movements. The SNAP Counting Units (SCUs) are the two elementary processes. For both processes, we are in a low complexity partition crossing, with 8 DETs; therefore, we can count 4*8=32 SNAP points for each of the SCUs, for a total of 64 SNAP Points (SPs).

In the end, we can measure the software impact to meet user requirement in analysis as 12 FPs and 64 SPs.

Example 2: dispatcher module

Consider a mobile App providing informative services to a Mobile Service Provider's customers. For the purpose



of this discussion, we focus on a service providing consumption rates.

The MSP ITC environment includes two distinct systems providing CRM services for Corporate and Consumer customers. The user requirement for the service in analysis is to get consumption rates from the proper system based on customer's mobile phone number (MSISDN), and provide it to the mobile App minimizing response time.

One of the problems to face in solution design is knowing whether the input MSISDN is Corporate or Consumer, in order to query the proper CRM system. In order to avoid, when possible, to query both CRM systems, design team introduce in solution architecture a local cache where the Consumer/Corporate information is stored after a successful CRM query. In such a way, on first access for an MSISDN it might happen to query both systems to get rates information, but on following accesses, it will be possible to query immediately the proper CRM system. Therefore, a dispatcher module is designed to check local cache before querying proper CRM system.

In FPA for this example, we can count a single Elementary Process, getTrafficData service, and a couple of EIFs for Corporate and Consumer rates information. Supposing 10 attributes read from the EIFs (without calculations) and sent back to the mobile user, we count a medium complexity EQ and two low complexity EIFs for 16 FPs.

Dispatcher and cache cannot be counted as functional components, but they are needed to meet the performance requirement. Therefore, we can apply SNAP and consider dispatcher and cache as a partition for the application in analysis. As for category 1.4 – Internal Data Movements, the SCU is the EP, it's a low complexity partition crossing with 2 DETs (MSISDN and typology), therefore, we can count 4*2=8 SPs. But we can consider also category 3.2 – Database technology for the local cache, a database object with non-functional purpose: the SCU is the EP, the complexity factor is low, and we have only one change, therefore we can count additional 6 SPs. Therefore, the additional effort needed for implementing dispatcher and cache can be measured in 14 SPs.

In the end, we can measure the software impact to meet user requirement in analysis as 16 FPs and 14 SPs.

Conclusion

We have analyzed two examples of combined use of FPA and SNAP in SOA architectures. In both cases, FP measurement may result too low to justify effective effort. But in both cases, there are some NFRs which can justify additional effort to produce clean, modular and efficient software. By using SNAP, these NFRs can be measured to better justify additional effort. With such combined use of FPA and SNAP, productivity expectations from software customers can be better satisfied.

REFERENCES

[1] R. Reggiani, G. Lanza, "Boundaries and SOA Architecture: a common issue", Milan, Sept 9th, 2014, GUFPI-ISMA Meeting

[2] IFPUG, "Software Non-Functional Assessment Process (SNAP) – Assessment Practices Manual, Release 2.3, May 2015

Certification Committee

By Greg Allen, Chair

July – December 2015

The Certification Committee kept busy in the last half of 2015 updating the CSP Exam to test on the most current release of the APM version 2.3. We have had CSP exams in Naples and Rome, Italy as well as one in Brazil. We have also had two regional CFPS Exams in Madrid, Spain and Tokyo, Japan.

January – June 2016

The committee will be reviewing presentations for CEP credit for regional conferences and holding more CSP Exams. The committee will be participating in a task force to examine possibilities improving and expanding the automation of exams. We will continue our ongoing work of reviewing exams, CEP submissions, and certification achievements to maintain the high standard associated with our certifications.

Greg Allen will be stepping down as Chairperson of the Certification Committee, a position he has held since April 2010. We wish to thank Greg for his hard work and dedication to the committee and IFPUG as a whole. Greg was able to maintain the integrity of our certifications while expanding our capabilities and member options. Greg will continue as a member of the committee and Lori Limbacher (Holmes) will be the new Chairperson.

Communications and Marketing Committee

By David Thompson, Chair

During the last six months of 2015, the Communication and Marketing Committee (CMC) focused on developing an IFPUG marketing plan, by adding material to the draft Marketing Strategy document first published in February, 2015. This included adding three short-term action items and three long-term action items. By the end of the year, work had started on the short-term action items, resulting in three drafted artifacts ready to move into production early in January: two case studies on the uses and benefits of function point analysis, and a shareable slide show on how to sell function point analysis to your manager.

Starting with a recommendation from Director Luigi Buglione, the CMC developed an updated set of Frequently Asked Questions (FAQs) and a mechanism to continually update them. The revised website page was moved into production on January 9, 2016.

In August, September, and October the CMC, through the website, the weekly eBlasts, and social media, engaged in a marketing campaign to publicize the ISMA[11] conference, classes, and exams held at the Blue Tree Premium Morumbi hotel in Sao Paulo, Brazil on November 17 and 18.

During the six month period, The CMC processed 38 web update requests and sent 26 eBlasts, on diverse topics, including those specifically targeted for the ISMA[11] conference. And in September, we began archiving eBlast copies on the website. Recognizing that more than 50% of our messages are read on mobile devices, we began using a new eBlast format "mobile-friendly" template

An additional accomplishment was posting, on the website, of one additional recorded webinar on a chapter in the newest IFPUG Book, The IFPUG Guide to IT and Software Measurement. These are linked from the website page, The Newest IFPUG Book.

And finally, we should mention the work to plan and produce the January 2016 edition of MetricViews.

Looking to the future, the committee will be continuing work on the IFPUG Marketing Plan, and will be implementing a process to periodically update the FAQs page.

Look for more on this in the first half of 2016!

Conference and Education Committee Report (CEC)

By Peter Thomas, Chair

The Conference & Education Committee (CEC) had a quiet second half of 2015, as the ISMA[11] and [12] conferences were organized, or being organized, by others.

We are working with other organizations for additional events and are seeking organizations who wish to partner with us. In particular, we would like to hold an event in India. Please contact cec@ifpug.org if your organization can help us.

We have begun preliminary planning for an event this fall and next year. Details will be published on the IFPUG website and sent via e-blast when we have them.

The CEC is continuing to publish a series of recorded webinars from the authors of the IFPUG Book, The IFPUG Guide to IT and Software Measurement.

Some are already available on the IFPUG website; others will be published in 2016.

You will find book purchasing information on the IFPUG website. There are forty-three chapters by fifty-two authors

from thirteen different countries, providing a comprehensive view on IT and Software Measurement.

Just a reminder, IFPUG members can access conference proceedings, at no charge, in the Knowledge Base within the Members Services Area of the IFPUG website.

We welcomed a new member to the CEC - Eduardo Alves de Oliveira - however we are still seeking additional members who will be available to attend the conferences. Send an email to ifpug@ifpug.org or complete the volunteer form.

Functional Sizing Standards Committee (FSSC)

By Dan French, Chair

2015 remained a busy and productive year for the IFPUG FSSC. In addition to our monthly committee meetings, the FSSC met for 3 days in April at ISMA[10] in Charlotte, NC, USA. During the conference, we worked on white papers, iTips, uTips and we did planning for the coming year. 2015 saw the committee publish a new iTip: iTip #7, Derived Data in Classifying an EO; and a new uTip #3, Early Function Point Analysis and Consistent Cost Estimating. We also published an update to iTip5, Real-time Data Sharing. In addition to the work completed this year on iTips and uTips, the FSSC also published a white paper: "Use of Function Points in an Embedded Systems-Stall Warning & Protection Computer (SWPC) System".

Currently, the FSSC is working on a number for projects slated for completion in 2016, including white papers on counting workflow applications; Universal Markup Language (UML) modeling; and an addendum to the Data Warehouse white paper released in 2007. The FSSC is also working on a joint white paper with the NFSSC. "Integrating Procedures for Function Point Analysis and SNAP."

Upcoming iTips include Counting Integrated EOs/EQs, and a collaborative project on Shared Data. The committee is also reviewing and updating previously released case studies to bring them into alignment with the current version of the Counting Practices Manual (CPM).

The committee would also like to thank Esteban Sanchez, Karl Jentzsch, and Carlos Eduardo Vazquez for their work in support of FSSC projects and looks forward to working with other dedicated IFPUG volunteers on future projects.

Non-Functional Sizing Standards Committee (NFSSC)

By Talmon Ben-Cnaan

SNAP method of non-functional sizing continues to evolve

The highlight of the second half of 2015 was a cooperation of COSMIC and IFPUG (represented by the NFSSC), to create a joined glossary of non-functional and project requirements. This document was published in both IFPUG and COSMIC sites after 6 months work and exchange of more than 20 iterations…

In addition, to the joint document, Charles Symons (COSMIC) and Talmon Ben-Cnaan (IFPUG) presented the results of the IFPUG-COSMIC collaboration for the Non-functional Requirements Glossary. The collaboration, as well as the presentation, was very well accepted by all participants.

Symons and Ben-Cnaan presenting

What's next?

NFSSC has started to collect data from users. Companies that measure SNAP and functions points report that the correlation of effort and size has improved compared to using function points only, and the software development effort is better explained by the combination of functional size and non-functional size.

However, the data that was sent to NFSSC shows three (out of fourteen) subcategories, in which the correlation between non-functional size and effort should be improved. NFSSC plans to analyze the data and improve the formulas of these subcategories.

The NFSSC will continue to support users by providing more examples and answering users questions.

In addition, NFSSC will continue to support growth of using SNAP. Recently, the APM was translated to Chinese; we hope to have users in other parts of the globe, looking for volunteers that will translate the APM to Japanese, Korean, Italian and more.

continued on page 22

(continued from page 21)

International Membership Committee Report

By Pierre Almen

The country representatives in the International Membership Committee, IMC, have continued to focus on answering requests from our members. In total, 126 requests from members in Brazil, China, India, Italy and Spain were handled during the last six months. This has been a much appreciated service for our members to communicate with a "local" contact. During autumn 2015, IMC has created a new type of membership, Junior membership that will be implemented next membership period starting July 1, 2016.

Upcoming activities include a review and improvement of the volunteer process and analyzing the requests we have received from our members to find out what can be improved in the current communication with our members.

ISO Standards Committee

By Steve Woodward, Chair

The ISO Standards Committee has had some structural changes, due to the recent IFPUG Board of Directors elections.

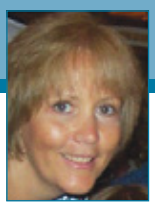
Congratulations to Carol Dekkers on being elected to the Board of Directors!

However, this, unfortunately, means that, due to IFPUG policies, Carol cannot continue as committee chair at this time.

I have accepted the chair responsibility, (with Carol's help), and look forward to moving forward with ISO and other standardization opportunities for IFPUG.

The ANSI and ISO SC7 (Software and Systems Engineering) meetings will continue, ensuring the IFPUG method has visibility and is integrated into the software standards best practices. The ISO Standards Committee also plans to take advantage, where opportunities exist, to further expand awareness and credibility of IFPUG in other standardization and software communities.

I look forward to 2016 and also plan to get more IFPUG members involved in the ISO Standards Committee activities to help ensure IFPUG is a recognized, trusted method of best practice for software sizing.



Message from the Past President

Kristie Lawrence, IFPUG
Immediate Past President

It has been my honor to serve the IFPUG membership over the past two years as President. I am now starting my two-year term as Immediate Past President and already have several tasks assigned to me to benefit both the membership and the smooth running of the Board of Directors.

Someone recently asked me about the changes that have occurred during my presidency. I have spent the last few days trying to think of what has occurred over this time. I thought I would provide a partial list of what the Board of Directors, the Committees, and our volunteers have done over the past two (2) years:

- Implemented the Countrywide membership level
- In the process of implementing the Junior membership level
- Implemented the SNAP Train the Trainer program, certified three (3) partners to provide this training, and have eliminated the need for IFPUG's Non-Functional Sizing Standards Committee to deliver the training
- Changed the makeup of the CSP exam questions to be 60% definition and 40% implementation
- Recognized ten (10) CFPS Fellows for twenty (20) continuous years of Certification
- Published several uTips, iTips, and even a vTip
- Published an updated SNAP Assessment Practices Manual (APM)
- Published several webinars based on IFPUG's most recent Guide to IT and Software Measurement
- Published the "Glossary of terms for Non-Functional Requirements and Project Requirements" a joint venture with COSMIC
- Held ISMA9 in Madrid, Spain so that the IFPUG Board could meet local members
- Provided ISMA10 in Charlotte, North Carolina, USA as a free conference to IFPUG members
- Provided greater value to IFPUG members by removing the fee for White Papers
- Increased IFPUG's investment portfolio thereby increasing IFPUG's balance sheet
- Held the 2015 Annual Meeting in Krakow, Poland so that the IFPUG Board could meet local members

Additionally, the IFPUG membership has changed the Board of Directors!

- There are more international members than members from the US (two years in a row)
- While not everyone will think this is a good thing, there are more women than men (two of the last three years)

I would like to thank the Board for their support throughout my tenure as President; Mauricio Aguiar, Christine Green, Dácil Castelo, Luigi Buglione, Pierre Almén, Joe Schofield, Tom Cagley, Debra Maschino, and Lori Holmes-Limbacher.

My desire is that my tenure as President leaves IFPUG a little better than it was. I continue to work for the membership through my service as the Immediate Past President. Most importantly, I hope to be able to meet many more of our members in Rome, Italy at ISMA[12] and wherever our future meetings are held.

CURRENT CONTACT INFORMATION?

To ensure you won't miss out on any IFPUG communications, please log in to your profile on the IFPUG Members Services Area and update your information.

Go to www.ifpug.org

Send emails to ifpug@ifpug.org,

call 609-799-4900 or fax 609-799-7032

Write to: IFPUG, 191 Clarksville Road, Princeton Junction, NJ 08550

Behind the Scenes

By Michele Giovine, Association Coordinator

Ever wonder who does what at IFPUG Headquarters? Let's check in with them and find out.

First let me introduce the IFPUG Headquarters Team. Connie Holden is the Executive Director, Nicole Lauzon is the Membership Coordinator and Michele Giovine is the Association Coordinator.

Connie Holden oversees the Association and works closely with the Board of Directors.

Nicole Lauzon responds to inquiries regarding membership, offers support for online purchases, processes the volunteer forms and provides them to the appropriate Committee chair. Nicole also takes care of the certification extension applications, so if your CFPS certification is expiring soon, be sure to expect an email from her. Don't forget your IFPUG membership needs to be current for an extension.

Michele Giovine handles all emails that come into ifpug@ifpug.org. She also processes all certification exam results. Michele would like to remind all members to review their personal information on the members' site to be sure it is up-to-date. Please list both personal and business email addresses.

Remember – in order to receive member benefits, including valid certification, your membership must be up to date. If you have any trouble with this process, please be sure to contact us at Headquarters right away!

We're very excited about this upcoming year, and we hope you're equally excited to be a member. As always, we look forward to hearing from you at ifpug@ifpug.org.

Documenting the Functional Size Measurement

By Carlos Eduardo Vazquez (carlos.vazquez@fattocs.com) and Guilherme Siqueira Simões (guilherme.simoes@fattocs.com)

Imagine taking your car to an auto repair center and they inform you that the total price is $2,000.00USD. Do you authorize the service under these terms?

It is not possible to make a good decision without knowing all of the items that will be reviewed and their values. If the car review includes only the oil change, you probably will not approve the budget because it will be too high considering only this service. However, if you are told you should change the oil, four tires, all shock absorbers and catalyst, maybe you will consider the value more reasonable and authorize the service.

The point to be emphasized is that the final function point value is not the total picture of the measured size. Providing the final measurement (or estimate) to the client is not enough; you must also provide all the rational used in the analysis. Otherwise, the client will not have enough information to check if the functional size presented is correct or not. Therefore, a common practice is to deliver the spreadsheet used to do the function point analysis to the individual using the measurement, allowing them to check it if needed.

The documentation is intended to add value to the measurement; simplifying an eventual audit and aiding to minimize the errors by the analyst responsible for the measurement. The documentation level of the analysis may vary (resulting in a greater or lesser effort in measurement). The level of detail in the documentation should be aligned to the purpose of measurement. For example, if the purpose is to estimate a rough order of magnitude of cost for the project, what is the meaning of a high detailed documentation? If the measurement does not have to be exact, the level of documentation does not have to be as thorough. **It is important to remember that one of the main goals of the measurement process is to be simple. Therefore, the level of documentation should be adjusted to balance the effort invested in measuring and include information that will add value to it.**

The level of documentation must be agreed between the parties in the measurement; balancing the costs and benefits involved. A high-level documentation measurement involves more time and cost. Each organization should set its documentation standards.

Although "Document and Report" are described in the latest step of the IFPUG counting process, it is executed at all steps of the process. The Counting Practices Manual highlights the following required items in the documentation:

Purpose and type of measurement

Measurement scope and application boundary

Measurement date

A list of all data and transaction functions, including their respective type and complexity, and the number of function points assigned to each function

Measurement result

Assumptions made and doubts resolved

And it suggests as optional items in the documentation (without intending to exhaust the subject):

Identification of the source documentation used for the measurement

A list of all participants of the measurement

Number of DETs, RETs and FTRs

A cross reference between data and transaction functions

A cross-reference between all functions and the respective requirements

Organization of Functions

Organizing the functions on a functional size measurement is crucial for its legibility. The easier it is to read, the lower the likelihood of errors during measurement (whether by omission or duplication of functions) and audit (if necessary). The basic principle is to fit together functions logically (from a business point of view). Generally, the analyst does not need to define this organization; the software or project documentation organizes the requirements according to this principle.

The analyst can use the following criteria for organizing its measurement functions:

User Manual: The index, chapters and sections follow a logical organization that can be used as a basis for organizing functions.

Menu System: For systems with an end user interface, the menu is the most appropriate, easy and direct approach for organizing functions.

Use Case diagram: Use cases provide another possible structure for the organization.

The above recommendations are only suggestions for an organization, but they do not apply to all cases. For systems without end user interface (no menu) the analyst should define a criterion.

Here is an example of organization for a timesheet system measurement.

Function	Type	DET	RET/FTR	Complexity	FP
Add justification	EI	5	2	Medium	4
Add standard working time	EI	4	2	Low	3
Check justification	EQ	5	2	Low	3
Check standard working time	EQ	4	2	Low	3
Employee (from HR)	EIF	3	1	Low	5
Import Justification data	EI	3	1	Low	3
Import standard working time data	EI	3	1	Low	3
Import Time Recording data	EI	4	1	Low	3
Justification	ILF	3	1	Low	7
List justifications	EQ	6	2	Medium	4
Login	EQ	4	1	Low	3
Modify justification	EI	5	2	Medium	4
Modify standard working time	EI	4	2	Low	3
Register Entry/Exit	EI	4	2	Low	3
Remove justification	EI	3	1	Low	3
Report justifications	EO	8	4	High	7
Standard Working Time	ILF	3	1	Low	7
Time Recording	ILF	4	1	Low	7
Timesheet	EO	10	4	High	7

Table 1: Timesheet system measurement organized in alphabetical order.

The previous example has an organized measurement, but it does not follow the basic principle of logical organization from a business point of view. It does not help with the document's legibility for those who need to understand the measurement.

Here are some guidelines for organizing the functions:

Segregate the data and transactions functions: As in the requirements analysis and design disciplines, in which specific artifacts are developed to document data and processes. It is common to use a data model (if available) to support the identification of logical files. Their organization in a specific section of the measurement (and not dispersed along with transactions) makes its reading easier.

Segregate ILFs and EIFs: When the number of logical files in the measurement is large (more than ten), organizing ILFs and EIFs into distinct sections improves the organization of measurement. It is common in a requirements specification to document external interfaces in specific parts of the document. To segregate ILFs from EIFs just follow the criterion for organizing requirements.

Segregate conversion functions: the transition requirements generally have a separate specification. As the conversion functions are not part of the application size, keeping them segregated in the project measurement facilitates updating the system size at the end of the project.

Grouping transactions by type (EI, EO and EQ) does not make sense from a business point of view; it can hinder the legibility of the measurement.

Consider the same measurement from the previous example now organized differently.

Function	Type	DET	RET/FTR	Complexity	FP
Files					
Time Recording	ILF	4	1	Low	7
Standard Working Time	ILF	3	1	Low	7
Justification	ILF	3	1	Low	7
Interfaces					
Employee (from HR)	EIF	3	1	Low	5
Transactions					
Login	EQ	4	1	Low	3
Time Recording					
Import Time Recording data	EI	4	1	Low	3
Register Entry/Exit	EI	4	2	Low	3
Timesheet	EO	10	4	High	7
Justification					
Import Justification data	EI	3	1	Low	3
List justifications	EQ	6	2	Medium	4
Add justification	EI	5	2	Medium	4
Modify justification	EI	5	2	Medium	4
Remove justification	EI	3	1	Low	3
Check justification	EQ	5	2	Low	3
Report justifications	EO	8	4	High	7
Standard Working Time					
Import standard working time data	EI	3	1	Low	3
Add standard working time	EI	4	2	Low	3
Modify standard working time	EI	4	2	Low	3
Check standard working time	EQ	4	2	Low	3

Table 2. Timesheet system measurement logically organized.

Note that it is not necessarily the analysts' responsibility to produce a measurement that is easier to understand. In this example, the reader may not see big differences. Small measurements (up to 100 PF, less than one-page document) are easier to understand, even though they may be poorly organized. The concern of the organization of functions becomes critical as the measurement size becomes larger.

Naming Functions

The nomenclature adopted for the functions also plays an important role in the documentation of the measurement. Suitable names also facilitate the measurement legibility and minimize the error-likelihood. Here's an example.

continued on page 26

(continued from page 25)

Function	Type	DET	RET/FTR	Complexity	FP
AACAPONT	ILF	4	1	Low	7
AAAHORPT	ILF	3	1	Low	7
AABJUSTT	ILF	3	1	Low	7
CDPESST	EIF	3	1	Low	5
GCBN90	EQ	4	1	Low	3
GCBNM0	EI	4	1	Low	3
GCBNM2	EI	4	2	Low	3
GCBNJ9	EO	10	4	High	7
GCBNJ2	EI	3	1	Low	3
GCBN64	EQ	6	2	Medium	4
GCBPK0	EI	5	2	Medium	4
GCBPK2	EI	5	2	Medium	4
GCBPK6	EI	3	1	Low	3
GCBPK8	EQ	5	2	Low	3
GCBPL0	EO	8	4	High	7
GCBPL2	EI	3	1	Low	3
GCBPRC	EI	4	2	Low	3
GCBPR1	EI	4	2	Low	3
GCBPR3	EQ	4	2	Low	3

Table 3. Timesheet system measurement using physical names.

This example is the measurement of the same system from the previous examples, with the same result. However, physical names of database tables and programs for the functions are used. This standard nomenclature damages the Function Point Analysis philosophy: abstracting from the software implementation. Thus, only those who know the system internals get to understand the measurement functions. This is not a good naming approach.

The basic principle for naming a function is that its name must be unique and representative according to the function it performs. It is not possible to have two functions with the same name in the same system or project. The documentation used for the analysis provides good tips in general to name the functions. Data models (logical or conceptual) or class models have entity or class names that are good options for naming logical files. The menu options, title of screens or reports, or use cases names also provide good tips for naming transactions. When this is not the case, the analyst must provide the most appropriate names.

One suggestion to name the logical file is to use nouns; a similar approach with the names of entities in a data model (for those who are familiar with data modeling). Do not use verbs in a logical file name so as not to confuse it with a transaction. Example: It is better to have an ILF called Customer than Store Customer. The latter option may lead one to think that it is the function that registers a new customer; which is not the case. Never use physical table names to name the logical files. Which is easier to understand: CDTB01 or Contract Off?

When the logical file has more than one subgroup, it is recommended to use a composed name that shows the subgroups that make up the file. For example: Order and Items, rather than Order. If the logical file has several subgroups, this method is not practical; the file name will be very large. In that case, use the simple name or a combination of the most important subgroups.

For EIFs, in addition to the previous guidelines, it is important to add a supplement to its name, reporting system data source. For example: "Customer (from CRM)". If a system has an interface with other, the name of the EIFs must show its source data. Otherwise, one would have to search for each of the EIF's in all interfaces in order to audit the measurement. It is recommended not to use the name of the source system to name the EIF because if there is the need to access multiple data sets, the EIFs must be counted separately and may not have the same name. For example, an EIF called HR only indicates that there is an interface with the HR system; while an EIF called Employee (from HR) is clearer.

One suggestion to name a transaction (when the documentation does not provide accurate options) is to use standards: infinitive verbs associated with an object on which the action is happening. For many cases the own verb defines the type of the transaction. For example, Client - Add (EI), Client - Modify (EI), Client List (EQ), Customer - Delete (EI), Customer - Consult (EQ), Customer, and Export (EQ / EO). **The use of program names for transactions is not recommended.** Which of these options is better for naming a process: KCB57 or Repay Bonds?

Summary

When documenting a function point count it is necessary to provide all the rational used in the analysis so the reader will have enough information to check if the functional size presented is correct or not.

The level of detail in the documentation should be aligned to the purpose of measurement. One of the main goals of the measurement process is to be simple. The level of documentation should be adjusted to balance the effort invested in measuring and include information that will add value to it. Each organization should set its documentation standards.

ISMA11 in Brazil
The IFPUG Conference returns to Brazil!

The annual IFPUG conference - International Software Measurement & Analysis (ISMA) - held its 11th edition in Brazil - São Paulo. This year the ISMA occurred together with the "Metricas 2015" (Brazilian Metrics Conference).

The conference - held on November 18th - hosted interesting and unique discussions, covering several aspects of measurement. The central theme was measuring the agile development process using function points.

Tom Cagley (IFPUG President) started the conference. He used his unique insight on commercial software development from his many years as a consultant to present the budget, estimation and planning differences between the methods advocated by the "#NoEstimates movement" and the classic estimate method.

After Tom, Steve Woodward presented "FPA in the cloud". Steve has been working with the Cloud community for much of its life and was able to share many in-depth insights in particular why agility and innovation is required to perform the counts.

Before lunch Luigi Buglione presented a comparison between agile and lean theory of measurement. Luigi has an extensive network of measurement specialists and he was an effective advocate of their experience.

After lunch Marcio Silveira presented what is important to analyze using function points when a software project is failing. Marcio is a member of the select group of IT Program Managers with many years of experience. This experience was reflected in his presentation.

Pierre Almén then presented problems in outsourcing projects without good quality metrics and how this situation could be avoided. Pierre has almost unique experience of benchmarking in the IT industry and was able to share his experience in his presentation using case studies.

Joe Schofield (via WebEx) used a Brazilian hat as a prop to show that it is possible to use function point and snap points in agile projects. His lack of physical presence did nothing to hide Joe's passion and experience.

Dácil Castelo presented productivity models using history bases from her company. Europe meets Latin America is part of Dácil's experience which enables her to share unique insights.

Finally, Kriste Lawrence presented how it´s possible for a company to begin operating with processes which deliver high quality using measurement and analysis. This is perhaps the most challenging aspect of measurement and it is made much easier when you can follow in the footsteps of others.

You can go to the IFPUG Member Services Area and login to access the ISMA11 presentations in the Knowledge Base.

Thanks to the "ISMA11 in Brazil" organizer: TI Metricas.

ISMA will come back to Europe next year. The 12th ISMA will be in Rome - Italy (May 3-5, 2016). For more information please click here.

See you in Rome!

IFPUG Board of Directors

Tom Cagley, President
DCG Software Value
t.cagley@softwarevalue.com

Mauricio Aguiar, Vice President
TI Metricas
mauricio@metricas.com.br

Christine Green, Secretary, Director of Certifications
Hewlett Packard Enterprise
christine.green@hpe.com

Debra Maschino, Treasurer
NASCO
debra.maschino@nasco.com

Kriste Lawrence, Immediate Past President
Hewlett Packard Enterprise
kriste.lawrence@hpe.com

Carol Dekkers, Director of Communications & Marketing
Quality Plus Technologies Inc
dekkers@qualityplustech.com

Dácil Castelo, Director of Sizing Standards
Leda-mc
dcastelo@leda-mc.com

Pierre Almén, Director of International & Organizational Affairs
ImproveIT
Pierrea@coolmail.se

Luigi Buglione, Director of Education & Conference Services
Engineering.IT SpA
luigi.buglione@eng.it

Committee Rosters

Certification Committee

- Lori Limbacher, Deloitte Consulting, LLP – ***Chair***
- Mahesh Ananthakrishnan, Cognizant Technology Solutions – ***Vice Chair***
- Gregory Allen, Pershing
- Joanna Soles, WellPoint
- Jim McCauley
- Teresa Beraldo, Banco Bradesco S/A

Communications and Marketing Committee

- Walter David Thompson, Blue Pine Solution Centre – ***Chair***
- Stephen Neuendorf, DCG Software Value
- Paul Radford, Charismatek Software Metrics
- David Herron, DCG Software Value
- Antonio Ferre Albero, GFT IT Consulting
- Justin Keswick, Bank of Montreal
- Carol Dekkers – Board Liaison
- Linda Hughes – VOLUNTEER

Conference and Education Committee

- Peter Thomas, Steria – ***Chair***
- Prof. Eduardo Alves De Oliveira
- Thiago Silva Da Conceicao, Synapsis
- Antonio Ferre Albero, GFT IT Consulting
- Toni Ramos, DCG Software Value
- Dr. Luigi Buglione – Board Liaison

Functional Sizing Standards Committee

- Daniel Bradford French, Cobec Consulting – ***Chair***
- Bonnie Brown, Hewlett Packard Enterprise – ***Vice Chair***
- Diana Baklizky, TI Metricas
- E. Jay Fischer, JRF Consulting
- Peter Thomas, Steria
- Adri Timp, Equens
- Roopali Thapar, IBM
- Tammy Preuss, AT&T
- Steve Keim, DCG Software Value
- Charles Wesolowski

International Membership Committee

- Pierre Almén
- Anjali Mogre, Atos Origin International SAS
- Cao Ji, Beijing Suiji Tech
- Gianfranco Lanza, CSI Piedmonte
- Ivan Pinedo, LDA Consulting, S.L.
- Saurabh Saxena, Amdocs Development Centre India Pvt Ltd
- Marcio Silveira, Hewlett Packard Enterprise

ISO Committee

- Steven Woodward, Cloud Perspectives – ***Chair***
- Carol Dekkers, Quality Plus Technologies, Inc.

Non-Functional Sizing Standards Committee

- Talmon Ben-Cnaan, Amdocs – ***Chair***
- Kathy Lamoureaux, Hewlett Packard Enterprise – ***Vice Chair***
- Mousa George Mitwasi, Optum
- Dr. Charley Tichenor
- Francisco Julian Gomez
- Tomasz Marchel
- Roopali Thapar, IBM
- Jalaja Venkat, iGATE Global Solutions
- Saurabh Saxena, Amdocs – VOLUNTEER
- Mauricio Aguiar, TI Metricas - VOLUNTEER
- Dr. Luigi Buglione, Engineering Ingegneria Informatica SpA – VOLUNTEER

Congratulations to these NEW and Extended Certified Function Point Specialists!

Marisa Accacio

Sandro Adanti

Fabio Aiello
Capgemini Italia SPA

Stefano Alunni
Capgemini Italia SPA

Lais Alves
TI Metricas Ltda

Angelo Amati
Engineering Ingegneria Informatica SpA

Stefano Arcangeli
Accenture

Maria Barbino
Capgemini Italia SPA

Marco Bassano
Capgemini Italia SPA

Rafal Bielicki
SAS Institute

Tiziana Borsini
IBM

Lucas Calmon
Banco de Brasilia S.A.

Mario Camilli
Capgemini Italia SPA

Gabriele Caramanica

Sabrina Cataldi

Angelo Celani

Maurizio Cesari
Capgemini Italia SPA

Przemyslaw Chelstowski
SAS Institute

Alessandra Ciolli
Hewlett Packard Enterprise

Angela Colagrossi
Indra Italia Spa

Roger Cordova

Fabio Costanzo
Capgemini Italia SPA

Andrea De Blasio
Engineering Ingegneria Informatica SpA

Giovambattista De Luca Carignani
IBM

Monica Del Buono
IBM

Nunzia Di Lecce
Capgemini Italia SPA

Dario Di Minno
Aruba SpA

Orietta D'Olimpio
IBM

Jonatas Dos Santos

Giuseppe Esposito
Capgemini Italia SPA

Jacopo Facchini
Aruba SpA

Isabella Fortino
Capgemini Italia SPA

Ferruccio Foti
Capgemini Italia SPA

Vincenzo Gagliarducci
Capgemini Italia SPA

Monica Gastalho

Maria Cristina Guantario
IBM

Tommaso Innocenti
Aruba SpA

Kumaresan Kandasamy
IBM

Jong Hyun Kim

Rajesh Koduru
MPHASIS

Makoto Kurashige
JFPUG-Japan Function Point Users Group

Rosangela La Fiandra
Capgemini Italia SPA

Andrea La Terra
Accenture

Marcieli Langer

Norma Liberati
ICE Agenzia

Annunziata Luciano
Sirti S.p.A.

Laura Massucci
TELECOM ITALIA INFORMATION TECHNOLOGY s.r.l .

Rohit Mehra

Ismael Melo
Abrantes Solucoes Ltda.

Stefano Meoni
Aruba SpA

Claudio Merolla
Capgemini Italia SPA

Clara Muccio
IBM

Ponmudi Mylsamy
Accenture

Denize Nabarro
Abrantes Solucoes Ltda.

Stefano Nigrelli
Aruba SpA

Fernando Oliveira
Banco de Brasilia S.A.

Irene Pace
Capgemini Italia SPA

Luz li Ibanez Paclieco
SOPRA GROUP INFORMATICA S.A

Loredana Paolini
Indra Italia Spa

Daniele Papa
Capgemini Italia SPA

Armando Parise
GEPIN PA SPA

Antonella Patrignani
IBM

Diana Pecelli
Engineering Ingegneria Informatica SpA

Cinthia Penetta Nunes

Giorgio Piccolo
Capgemini Italia SPA

Paolo Pioli
Capgemini Italia SPA

Valentina Pitorri
Engineering Ingegneria Informatica SpA

Maria Enza Irma Policicchio
Capgemini Italia SPA

Aldo Pondaco Neto
Accenture

Ponrathi Ponpandian
IBM

Veronica Porta
Sistemi Informativi SpA

Prasanna Raja
IBM

Tamara De San Teodoro Rodrigo
LEDA Consulting, S.L.

Roberta Russo
Hewlett Packard Enterprise

Tirupati Sahu
IBM

Claudia Salgues
Tribunal Regional Federal Da 5a Regiao

Stefano Salmeri
Indra Italia Spa

Luca Santillo
Agile Metrics

Cristina Scafetta
Indra Italia Spa

Guido Servili
Capgemini Italia SPA

Damian Sierajewski
Accenture

continued on page 30

(continued from page 29)

Monica Silva
Superintendencia De Seguros Privados

Marcia Silva De Morais
CTIS Tecnologia S/A

Roberta Straneo
Indra Italia Spa

Ryo Takahashi
JFPUG-Japan Function Point Users Group

Francesca Tollis
Indra Italia Spa

Francisco Javier De La Cuesta Torres
INDRA Sistemas, SA

Paulo Tortorelli

Adriana Trentini
TI Metricas Ltda

Aruna Valeti
IBM

Rosana Veras Paulino

Giorgio Veroi
Indra Italia Spa

Stanislao Vigna
Capgemini Italia SPA

Saurabh Wani
ACCENTURE

Hyong Kyun Yang
LG CNS

Kiran Yeole
Amdocs Development Centre India Pvt Ltd

Congratulations to these NEW Certified Function Point Practitioners!

Kumar Amrit
ACCENTURE

Samuel Batagliao
Accenture

Poushali Bhadra
IBM

Maria Cristina Cadolini
TELECOM ITALIA INFORMATION TECHNOLOGY s.r.l .

Elena Biglino Campos
INDRA Sistemas, SA

Lucas Cavalcanti

Eleonora Cesaretti
Eustema S.p.A

Zelia Costa

Maria Carolina De Menezes

Kanchan Dhar
CGI Group Inc

Sonia Fraioli
Eustema S.p.A

Jose Menendez Garcia
Atos Spain SA

Dong Pill Kim

Sunil Kumar

ACCENTURE

Dario Mencarini
Indra Italia Spa

Francisco Neto
PITANG CONSULTORIA E SISTEMAS S/A

Pietro Nico
IBM

Sandro Politi
ICE Agenzia

Cassiano Santana

Juscicleide Santos
SoftNet

Joao Gabriel Santos
Eficácia Organização

Jussania Souza

Alan Tome De Souza
ATSNET Informática e Desenvolvimento de Sistemas LTDA – ME

Sabina Wolski
TELECOM ITALIA INFORMATION TECHNOLOGY s.r.l .

Congratulations for 20 years of CFPS Certified Function Point Fellows!

Fall 2014

Loredana Fallicciardi,
DDWAY S.R.L.

Steve Neuendorf,
David Consulting Group

Spring 2014

Mary Dale,
Q/P Management Group, Inc.

Roger Heller,
Q/P Management Group, Inc.

Debra Maschino,
NASCO

Mousa George Mitwasi,
Optum

Bruce Paynter,
BNB Software Quality Management

Bruce Rogora,
Pershing, LLC

Joanne Soles,
WellPoint

Andrew Sanchez

Steve Woodward,
Cloud Perspectives

Congratulations to these NEW Certified SNAP Practitioners!

Elson Alves Junior
TI Metricas Ltda

Cristiane Barroso
DIGI SYSTEM

Cintia Aguiar Batista
TI Metricas Ltda

Chiara Broccia
Shared Service Center S.R.L.

Viviana Carla Dimas Hirama
TI Metricas

Giuseppe Imparato
Engineering Ingegneria Informatica SpA

Diego Emanuel Ferreira Da Rocha (Diego Da Rocha Ferreira)
BANCO BRADESCO S/A

Marcos De Freitas Junior
TI Metricas

Luiz Gustavo Queiroga Pena
Caixa Economica Federal

Christiano Poiato
Indra Brasil Soluções e Serviços Tecnológicos S/A

Marcelo Elias Nunes Ribeiro
TI Metricas Ltda

Ismael da Silva de Melo
Confederação Interestadual das Cooperativas Ligadas ao Sicredi

Giovanna Perin
TELECOM ITALIA INFORMATION TECHNOLOGY s.r.l.

Gustavo de Oliveira Santos
Everis Brasil Consul de Neg & Tec. da Inf. Ltda

Isabela Del Corso
TI Metricas Ltda

Marcos Eduardo Neves
TI Metricas Ltda

Maria Paola Rapanotti
Engineering Ingegneria Informatica SpA

Filipe Silva
TI Metricas

Danilo Santos
TI Metricas Ltda

Luana Coelho Texeira
TI Metricas

Marcello Tortora
Engineering Ingegneria Informatica SpA

Fabrizio Valiani
Engineering Ingegneria Informatica SpA

Rodrigo Vidal
TI Metricas Ltda

Luciano Zu
Engineering Ingegneria Informatica SpA

Certification Matters!

Kunal Punjabi, Belgium

"Being an IFPUG CFPS is like getting a valid visa to the world of software estimation and measurement. It enhances client credibility and earns you mutual respect among the senior management stakeholders. Your colleagues look up to you, and appreciate and seek your guidance. And it strengthens your understanding in working with functional and nonfunctional requirements."

Sridevi Devathi, India

"My IT Career got transformed as I did CFPS certification 9 years ago. It opened many avenues – I was immediately asked to create FP COE, which evolved into corporate Estimation COE. Got to interact with estimation experts, pursue research, training and consulting in Estimation and Benchmarking."

Giovanni D'Alessandro

"I have dealt with metrics 18 years ago for c.a. 7 years and after a long break I started six years ago on non-standard techniques. Passing the certification strengthened my knowledge, opened new mental schema, rejuvenated my approach to the subject and revitalized my commitment."

Raffaele Russo

"Becoming a CFPS was an important step in my professional career path . In my company (Unisys Italy) I'm the Focal Point for Function Point Analysis, Productivity, Effort Estimation and Software Measurement. I'm accountable for counting Function Points (IFPUG 4.3) for Public Administration Customers."

191 Clarksville Road
Princeton Junction, NJ 08550
USA

The 12th IFPUG International Software Measurement & Analysis Conference

"Creating Value from Measurement"

May 3-5, 2016 – Rome, Italy

Coming back to Italy after 20 years, this new edition of the IFPUG ISMA Conference will provide a forum for practitioners and researchers to discuss most recent advances in planning and sustaining measurement programs from both practical and theoretical perspectives in the scope of software value creation and value-based management in software product and service development organizations. We invite professionals responsible for, involved in, or interested in software measurement to share innovative ideas, experiences, and concerns within this scope.

ISMA 12 is organized by:

GUFPI ISMA

on behalf of: