Bidirectional Influence of Defects and Functional Size

Sylvie Trudel
Pyxis technologies
Laval, Canada
e-mail : strudel@pyxis-tech.com

Alain Abran
Dept. of Software Engineering and Information Technologies
École de Technologie Supérieure – Université du Québec
Montreal, Canada
e-mail: alain.abran@etsmtl.ca

***Abstract*— This paper reports on a research project investigating the contribution of functional size measurers to finding defects in requirements. In previous experiments, the concurrent inspection of the same requirements document by a measurer in addition to inspectors showed that the functional size measurement activity adds value in finding functional defects that inspectors have not identified. This paper presents an analysis of the influence of defects on the functional size once many of the identified defects have been fixed. For these experiments, the measurers used COSMIC – ISO 19761 to measure functional size and to find defects. Experiments findings show an increase in functional size and a decrease in defects identified.**

Keywords- Functional size, COSMIC – ISO 19761, FSM, measure, quality, defects.

I. INTRODUCTION

A software requirement is a property which must be exhibited by software developed or adapted to solve a particular problem. Requirements generally fall into two categories: functional and non-functional. Functional requirements describe the functions that the software is to execute; for example, formatting some text or modulating a signal. Nonfunctional requirements are the ones that act to constrain the solution [1]. Functional requirements describe the software's functionalities while non-functional requirements, also called technical and quality requirements, describe the software's attributes such as performance, security, and reliability. The research work reported here focuses on functional requirements.

During the early phases of a software development life cycle, the requirements documents are used as inputs to the estimation process, including for measuring the functional size of the software to be developed. The quality of a requirements document is therefore important, and will impact the consistency of the measurement results as well as the confidence in the estimation outcomes.

Requirements impact all phases of the software life-cycle as shown in Figure 1. Therefore, ambiguous, incomplete and incorrect requirements may negatively impact all phases if not detected early enough to be corrected; when not found, those defects will typically require rework to rectify work done in previous phases of the life cycle.

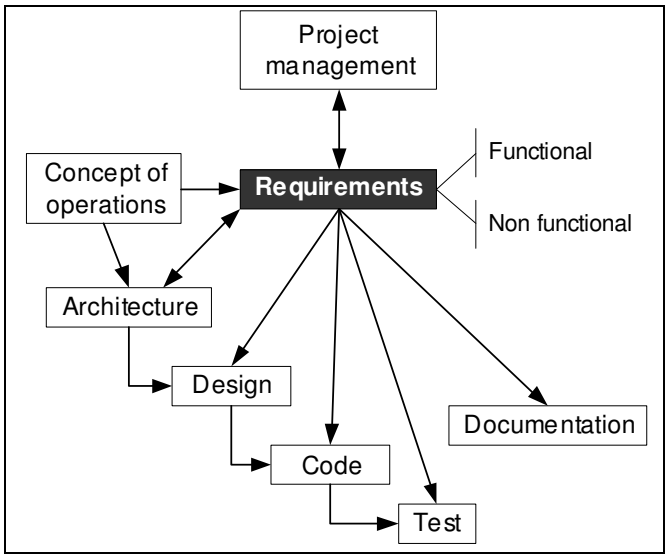

Figure 1. Requirements usage in software development life-cycle phases.

To minimize rework effort and cost for fixing defects at later phases in the development life-cycle, many organizations apply various review techniques on their requirements documents. Review techniques typically include a set of rules to help requirements authors and reviewers in achieving quality attributes of their requirements, such as those stated in the IEEE 830 [2]: "Correct", "Unambiguous", "Complete", "Consistent", and "Verifiable".

An inspection [3] is a review technique known to be efficient at identifying defects but, like any other review technique, it does not guarantee that all defects are found. To increase the efficiency and effectiveness for finding defects in software artefacts, it is recommended that organizations use several verification techniques. Review efficiency represents the ability of a software team to identify and remove defects in an artefact. Review efficiency can be measured in number of defects found in that artefact at

review time compared to the total number of defects found in the whole software project for which the origin can be traced back to that same artefact. Review effectiveness corresponds to the average effort spent in identifying critical defects.

In the early phases of the development life cycle, these same requirements documents are also used as an input for the measurement of the software functional size, typically for estimation purposes. When carrying this measurement process for estimation purposes, measurers often observe a number of defects in the functional requirements. This contribution of measurers at finding defects in requirements documents has been investigated in [4][5], and a summary of the results are presented in section II of this paper.

A. *The COSMIC method*

Functional size measurement (FSM) is a means for measuring the size of a software application, regardless of the technology used to implement it.

The COSMIC FSM method [6] is supported by the Common Software Measurement International Consortium (COSMIC) and is a recognized international standard (ISO 19761 [7]). This specific FSM method was chosen and applied in the experiments described in this paper.

In the measurement of software functional size using COSMIC, the software functional processes and their triggering events must be identified. The unit of measurement in this method is the data movement, which is a base functional component that moves one or more data attributes belonging to a single data group. Data movements can be of four types: Entry (E), Exit (X), Read (R) or Write (W). The functional process is an elementary component of a set of user requirements triggered by one or more triggering events, either directly or indirectly, via an actor. The triggering event is an event occurring outside the boundary of the measured software and initiates one or more functional processes. The sub processes of each functional process constitute sequences of events, and a functional process comprises at least two data movement types: an Entry plus at least either an Exit or a Write. An Entry moves a data group, which is a set of data attributes, from a user across the boundary into the functional process, while an Exit moves a data group from a functional process across the boundary to the user requiring it. A Write moves a data group lying inside the functional process to persistent storage, and a Read moves a data group from persistent storage to the functional process. See Figure 2 for an illustration of the generic flow of data groups through software from a functional perspective.

The balance of this paper is organized as follows. Section II presents an overview of the previous experiment. Section III presents the experimental steps of the experiment reported in this paper. Section IV presents the analysis of the data from this experiment, and Section V presents a discussion of findings and future work.

II. PREVIOUS EXPERIMENT

A. *Purpose and objective of the previous experiment*

The main objective of the previous experiment was to assess the efficiency and effectiveness of the COSMIC method as a method for finding defects in software functional requirements.

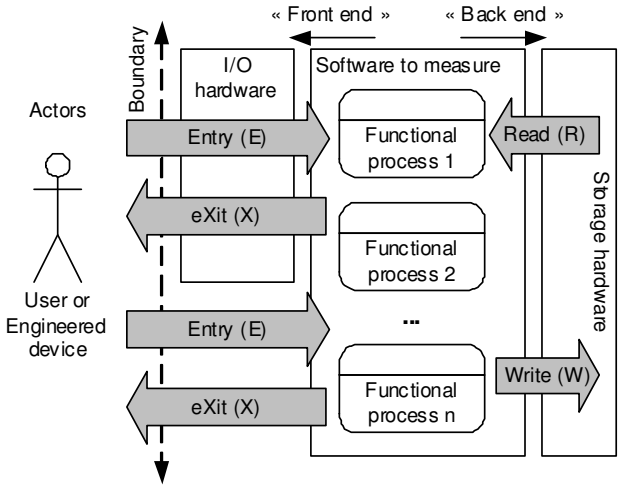

Figure 2. Generic flow of data through software from a functional perspective.

The purpose was to perform an experiment involving experienced practitioners, some of whom would be skilled in measuring functional size with the COSMIC method and others who would either be skilled in inspecting requirements or be knowledgeable on what is a well written software functional requirement. Special care was taken to get experienced practitioners in FSM and experienced inspectors and requirements writers in participating to this experiment. Four inspectors and five measurers participated in previous experiment.

B. *The SRS document and its evolutions*

1) *Description of the SRS*

The software requirements specification (SRS) document that was chosen for the experiment was compliant with IEEE 830 for its structure and content. This SRS was also compliant with UML 2.0 [8] for the use case diagram, the behavioral state machine, and use case details.

The SRS was entitled “uObserve Software Specification” [9] and had 16 pages of descriptive text in English of approximately 2900 words. “uObserve” is an event-driven system composed of two software applications running on different PCs and exchanging data over a Local Area Network (LAN).

Section 1 of the SRS describes the introduction, purpose and scope, project objectives, background information, and references. Section 2 provides a high-level description of the system to develop, the list of features and functions (included and excluded), user characteristics, assumptions, constraints,

and dependencies. Section 3 lists all specific requirements, beginning with the user interface and its prototype, the hardware interfaces, followed by functional requirements (section 3.2), and quality requirements (section 3.3).

Functional requirements included a use case model diagram with 12 use cases but only 10 use cases had detailed descriptions. This issue was later raised as a defect by several participants.

2) Evolution steps of the SRS document

The uObserve software described in the SRS was meant as a proof of concept system for a usability testing tool applying client-server architecture. It was developed in Java and successfully tested in 2004 by a team of two developers. The initial version of the SRS had been written by one the developer, peer reviewed by the other developer, and approved by the project sponsor.

Few years later, when this SRS document was chosen for these experiments, it was also reviewed by an experienced practitioner for defects and was labeled "v1.0" once defects were fixed. This version 1.0 of the SRS was used in the previous experiment.

C. Experiment steps

The steps of the previous experiment were as follows:

1. The experiment was prepared by the experimenter:
   a. Material was prepared: SRS, training material, data capture forms;
   b. Call for participation was done: experienced practitioners were invited in a conference workshop;
2. Training was provided by the experimenter to inspectors and measurers on the chosen inspection method so they would all apply the same set of rules, roles, and, most importantly, the same defect types and categories;
3. The inspection method was applied while objectively supervised:
   a. The inspection was planned (10 min.);
   b. A kick-off meeting was held (5 min.);
   c. Inspectors performed their individual checking;
   d. In parallel of step 3c, experienced measurers did their FSM activity while identifying functional defects and issues;
   e. A "logging meeting" was conducted by the experimenter with all participants to gather defect data and reach common understanding of those defects;
4. Experiment data was compiled and logged in a spreadsheet by the experimenter:
   a. Defects and issues;
   b. FSM detailed data;
   c. Individual effort.
5. Experiment data was reviewed by participants to ensure accuracy: scanned copies of their manuscript notes were sent to them along with their logged data so they could compare the logged data with their notes and report any difference or precision;
6. Experimental data was analyzed once its quality was assured.

D. Summary of results from previous experiment

The functional descriptions in section 3.2 of the SRS are of primarily importance for the measurement activity while content of other sections is informative with regards to FSM activity. For this reason, results presented here focus on functional requirements.

1) Defect and issue data

Defects and issues were categorized as "Functional" (F) and "Non-functional" (N). Defects were assigned a defect type by participants: Critical or major (C), Minor (M), and Spelling/Syntax (S). Issues were either a Question (Q) to the SRS author or an Improvement suggestion (I).

A total of 274 individual defects and issues were logged from measurers and inspectors notes. However, many defects were identified as duplicates during the data analysis phase leaving a total of 194 uniquely identified defects (TABLE I) and 34 uniquely identified issues (TABLE II).

TABLE I. DEFECT DATA PER CATEGORY AND TYPE.

Defects found by	**Functional**			**Non-funct.**			**Total**
	C	**M**	**S**	**C**	**M**	**S**	
Inspectors only	23	46	5	14	14	6	**108**
Inspectors and measurers	4	1	3	3	3	4	**18**
Measurers only	17	18	11	5	5	12	**68**
Sub-total:	**44**	**65**	**19**	**22**	**22**	**22**	**194**
Total:	**128**			**66**			

TABLE II. ISSUE DATA PER CATEGORY AND TYPE.

Issues found by	**Functional**		**Non-funct.**		**Total**
	Q	**I**	**Q**	**I**	
Inspectors only	2	9	0	12	**23**
Inspectors and measurers	0	0	0	0	**0**
Measurers only	7	1	1	2	**11**
Sub-total:	**9**	**10**	**1**	**14**	**34**
Total:	**19**		**15**		

In summary, measurers found 79 defects and issues (68+11) that inspectors had not identified, during the same elapsed time, while also providing the functional size of the system. The analysis of defect data demonstrated a value-added between 15% and 32% in defects found by any single measurer over all inspectors together [4]. These results show the influence of FSM on finding defects.

2) Functional size data

Functional size measurement results in COSMIC Function Points (CFP) showed some variations among measurers, ranging from 51 CFP to 62 CFP, with an average size of 57 CFP and a standard deviation of 4.5.

Some of these size variations may be due to defects in the SRS. Other sources of variations are discussed in section V.

3) Effort data

Measurers spent 58 minutes on average to measure and find defects and issues, with a standard deviation of 12 minutes. The average relative effort was 1.0 min/CFP with a standard deviation of 0.2.

Inspectors spent 59 minutes on average to find defects and issues, with a standard deviation of 5 minutes. We

considered the level of effort to be equivalent between measurers and inspectors.

III. THIS EXPERIMENT

This paper reports on a similar experiment than those described in section II, but using an improved version of the SRS document, in which a significant portion of defects and issues identified in the previous experiment have been fixed.

A. Objective of this experiment

The objective is to measure the effect of having fixed the majority of identified defects on the functional size of the software.

The mean was to perform an experiment involving experienced practitioners skilled in measuring functional size with the COSMIC method.

B. Experiment steps

1) Prepare the experiment

For this experiment, it was necessary to prepare an improved version of the SRS: after the completion of the first experiment, the uObserve SRS was edited by the experimenter to fix most of identified defects (84%, see TABLE III) and address most of issues (88%, see TABLE IV).

Three status descriptions were defined and applied in the defects and issues log:

- Solved: specific modifications to the SRS solved directly the defect or issue.
- Closed: solving specific defects or issues has made this defect or issue obsolete, therefore solved as a side-effect. It was deemed important be able to make a distinction between defects and issues directly or indirectly solved by modifications for data analysis.
- Open: the defect or issue has not yet been fixed or addressed.

All defects found by measurers were either solved or closed. Applied modifications were reviewed against the defect list as well as the original version (v1.0) before issuing the new SRS version (v2.0) [10].

TABLE III. DEFECTS FIXING STATUS PER CATEGORY AND TYPE FROM V1.0 TO V2.0 OF THE UOBSERVE SRS.

Defects status	Functional			Non-funct.			Total	%
	C	M	S	C	M	S		
Solved	38	53	17	11	14	16	**163**	**84%**
Closed	3	5	0	1	3	2		
Open	3	7	2	10	5	4	**31**	**16%**
Sub-total:	**44**	**65**	**19**	**22**	**22**	**22**	**194**	**100%**
Total:	**128**			**66**				

TABLE IV. ISSUES ADDRESSING STATUS PER CATEGORY AND TYPE FROM V1.0 TO V2.0 OF THE UOBSERVE SRS.

Issues status	Functional		Non-funct.		Total	%
	Q	I	Q	I		
Solved	5	8	0	9	**30**	**88%**
Closed	3	0	1	4		
Open	1	2	0	1	**4**	**12%**
Sub-total:	**9**	**10**	**1**	**14**	**34**	**100%**
Total:	**19**		**15**			

Defects and issues that had an open status for v2.0 required a significant level of effort to find a suitable solution. It was decided to pursue with this experiment using nonetheless the significantly improved SRS document on the basis that the perfect SRS might not exist or may require unreasonable effort to obtain.

The new v2.0 SRS contains 15 use cases in the use case model diagram. All of these use cases are described in details in the functional requirements section of the SRS (the same 3.2 section as in v1.0 of the SRS).

Also, a measurement procedure was prepared to ensure that all participants apply the same steps and the same expected effort (Figure 3).

Your mission is to measure Functional Processes (FP) (section 3.2 only of the uObserve SRS v2.0) with the COSMIC method while identifying defects and issues (if you notice any). Measurement and defect identification should take between 1.0 and 1.5 hour to complete.

1. Use the revision mode in MS-Word to measure by highlighting FPs, triggering entries, and Data Groups (DG), then insert a comment and simply indicate movements (EXRW). The experimenter will cumulate this data.
2. Use comments also to indicate any defect, question or assumption that may or may not be relevant to measurement.
3. Send back the commented file with the effort spent.

Figure 3. Measurement procedure of the experiment.

2) Invite experienced measurers

The same five experienced measurers were invited to measure the functional size of the uObserve SRS v2.0. The document was emailed to them along with the experimental measurement procedure. All five invited experienced practitioners accepted to participate in this experiment using the procedure.

3) Compile data from participants

The following data was received from participants and compiled into a spreadsheet:

- Their annotated copy of uObserve SRS v2.0, which contained comments indicating defects, issues, and measurement assumptions as well as detailed FSM results.
- Effort spent applying the experiment measurement procedure.

4) Objectively ensure accuracy of compiled data

A colleague independently verified that written notes from participants were adequately captured in the experiment spreadsheet. Those written notes contained defects, issues, assumptions, and measurement data. Also, he verified that the effort sent by participants by email was adequately captured.

5) Analyze data

Data from this experiment was analyzed and compared to data from the previous experiment.

C. Experiment results

1) Defects, issues, and assumptions data

Altogether, measurers found 11 defects and 5 issues in v2.0 (see TABLE V). These numbers are significantly smaller than what was found by measurers in v1.0 (86 defects and 11 issues). No duplicate were observed among these items.

TABLE V. DEFECTS, ISSUES, AND MEASUREMENT ASSUMPTIONS, PER CATEGORY AND TYPE, AS NOTED BY MEASURERS.

Type	Defects			Issues		A
Category	**C**	**M**	**S**	**Q**	**I**	
Functional	4	5	0	2	2	10
Non-functional	0	1	1	1	0	10
Sub-total:	**4**	**6**	**1**	**3**	**2**	**20**
Total:	**11**			**5**		

2) Functional size data

The uObserve SRS v2.0 has 15 use cases (UC) descriptions with identified triggering events and divided into two distinct software applications.

One measurer assumed that the functional decomposition was at the appropriate level after studying every UC and applying the COSMIC rules and principles to his understanding of the system. Another measurer assumed that almost every event was a triggering event for a new functional process. Their assumptions had low effect on their size results but a significant effect on the number of functional processes.

But measurer #5, who mentioned not having applied the COSMIC method for a couple of years, did not see that the two applications were users of each other, on top of both having the same human user. Based on the assumption that there was only one single human user for the whole system, he considered that those UCs not involving that human user should be ignored when measuring because he considered them as being implementation UCs. This wrong assumption led this measurer to apply the COSMIC method on half of the average number of functional processes identified by the other measurers. For that reason, size and effort data related to v2.0 for this measurer were excluded from average and standard deviation calculations (see TABLES VI and VII).

TABLE VI. NUMBER OF FUNCTIONAL PROCESSES IDENTIFIED.

Measurer	Functional processes		
	v1.0	**v2.0**	**Difference**
#1	11	12	+1
#2	10	15	+5
#3	12	21	+9
#4	10	14	+4
#5 (see note)	11	8	-3
Average:	**10.8**	**15.5**	**4.8**
Std.dev.:	**0.8**	**3.9**	**3.3**

Note: v2.0 data excluded from average and std.dev. due to wrong measurement assumption.

TABLE VII. FUNCTIONAL SIZE.

Measurer	Functional size (CFP)		
	v1.0	**v2.0**	**Difference**
#1	61	68	+7
#2	51	97	+46
#3	57	81	+24
#4	62	71	+9
#5 (see note)	55	33	-22
Average:	**57.2**	**79.3**	**21.5**
Std.dev.:	**4.5**	**13.1**	**18.0**

Note: Size excluded from average and std.dev. due to a wrong measurement assumption.

3) Effort data

TABLE VIII shows the effort (in minutes) and the relative effort (in minutes per CFP) spent by each measurer, with the average and standard deviation.

TABLE VIII. ABSOLUTE EFFORT AND RELATIVE EFFORT SPENT BY MEASURERS.

Measurer	Effort (minutes)	Size (CFP)	Relative effort (minute/CFP)
#1	90	68	1.3
#2	240	97	2.5
#3	90	81	1.1
#4	76	71	1.1
#5 (see note)	90	33	2.7
Average:	**117**	**79**	**1.5**
Std dev.:	**69**	**13**	**0.7**

Note: Relative effort and size excluded from average and Std.dev. due wrong measurement assumption.

Compared to the effort required to measure v1.0, the effort to measure v2.0 was 102% higher on average. The relative effort was 50% higher on average with a standard deviation three times higher, when excluding data from measurer #5.

Measurer #2 reported his effort result significantly higher than all other measurers, mentioning that he took time at the first reading of the SRS to ensure all use cases were indeed functional processes.

IV. ANALYSIS

A. Measurement assumptions and effect on results

Differences in measurement results can all be explained with assumptions that were explicitly noted by measurers. In industry, measurers usually have access to the SRS author or one of the project' software engineers to verify their assumptions against the reality before completing the measurement. This step has not occurred in this experiment. Therefore, when a wrong assumption is made, the corresponding functional size is also correspondingly inaccurate.

Documenting measurement assumptions is essential in order to verify and compare measurement results. Important differences are likely to occur when wrong assumptions are made on:

- The **level of decomposition**: when measuring the software size of any system, it is of primary importance that the level of decomposition be at such a level that the software boundaries become

visible. Otherwise, the measurement exercise becomes meaningless as it does not measure software but an abstraction of one or many different software composing that system.

- Identified **boundaries**: many event-driven systems are composed of more than one application, each of which may be a functional user candidate of another. Adequately identifying the software boundaries is directly linked with the assumed "level of decomposition". When assumed to be at the "system" level, the software boundaries may not be visible. Any wrong assumption in that matter has an effect on the measured size and the number of identified functional users.
- Identified **functional users**: human users are the easiest to identify. Even when software boundaries are adequately identified, it is possible that some functional users may be missed, in particular when the functional documentation is unclear about interfaces. Any missed functional user (peer software application, scheduler, or hardware device) has a direct impact on the measured size.
- The **functional processes**: when applying the COSMIC method measurement procedure, triggering events must be identified to adequately identify functional processes. Measurement of event-driven software applications is particularly sensitive to this step. Expert measurement results of SRS v2.0 showed significant differences in the number of functional processes, mainly for stating different assumptions on whether an event is considered triggering a separate functional process or is part of an existing functional process.
- The **data model**: information systems documentation often provides a data model that is used as an input for identifying data groups and related data movements. Data models can take many forms, such as an entity-relation diagram or a class diagram. Still, data models may provide all required information to clearly identify Read and Write data movements, but few or no information on Entry and eXit data movements related to interfaces with the identified functional users. Any wrong assumption on the data model has an effect on the functional size.

B. *Effect of defects on FSM results*

With the majority of defects fixed in v2.0, the average functional size has increased 39% on average. This significant difference is largely caused by requirements that were clearer, more complete, and consistent. Therefore, the results suggest that these defects had an influence on the functional size.

C. *Significantly fewer defects and issues*

It was expected that a significantly smaller number of defects would be found while measuring uObserve SRS v2.0 in comparison to the number of defects found in v1.0. Some participants stated they found only few defects. One measurer stated that he did not have the time to note more defects. There is a possibility that measurers would have found more defects if allotted more time.

D. *Sub-optimized mechanism applied*

Applying the COSMIC FSM method using the revision mode in MS-Word might not have been the best optimal measurement mechanism, as noted by some measurers. Handwriting measurement on a printed copy of the SRS would have given them with more time to identify defects. On the other hand, taking these notes directly in an electronic file simplified the verification of measurement results, defects, issues, and assumptions.

E. *Threat to validity*

Experiments described in this paper compiled data from a relatively small number of participants. Results and analysis could be different with a larger sample. Measurement results from experiments with inexperienced measurers on the same SRS (v1.0 and v2.0) showed that these new practitioners did not apply the COSMIC method measurement procedure adequately, and could not be included in the research reported here.

V. DISCUSSION AND FUTURE WORK

In metrology, it is well accepted that to each measurement results there is an associated measurement uncertainty [11]. This seems to be especially the case when dealing with event-driven software applications, due to the unfamiliarity of those accustomed to measure traditional management information application. Measurers of event-driven applications face many challenges, such as adequately identifying the applications boundaries, their functional users, their functional processes, and their data groups. If such applications are not using a database, their SRS are not likely to contain a data model making it more challenging for measurers to come up with accurate and repeatable functional size measurements results.

At the time this paper was being written, a new COSMIC guideline was being drafted to help improve repeatability when measuring real-time applications. We recommend to include the findings of this experiment in the future COSMIC document to guide any COSMIC practitioners in applying the rules and principles.

Future work includes addressing remaining known SRS defects and issues in order to publish v3.0 of the uObserve SRS as an agreed measurement case study.

ACKNOWLEDGMENT

We are grateful to expert measurers who generously participated in experiments described in this paper: Jean-Marc Desharnais, Luca Santillo, Charles Symons, Harold van Heeringen, and Frank Vogelezang.

REFERENCES

[1] A. Abran, J.W. Moore, P. Bourque, R. Dupuis, Guide to the Software Engineering Body of Knowledge (SWEBOK). IEEE Computer Society: Los Alamos, CA, 2004; 202 p. On Line at: http://www.swebok.org.

[2] IEEE Computer Society, IEEE-Std-830-1998, IEEE Recommended Practice for Software Requirements SpeciÞcations, New York, NY, June 1998.

[3] K. Wiegers, Peer Reviews in Software: A Practical Guide, Boston, MA: Addison-Wesley, November 2001.

[4] S. Trudel and A. Abran, "Improving quality of functional requirements by measuring their functional size", International Workshop on Software Measurement -IWSM 2008, Metrikon 2008, and Mensura 2008, Springer, Munich, Germany, Nov. 2008, pp. 287-301, ISBN 978-3-540-89402-5.

[5] S. Trudel and A. Abran, "Functional Size Measurement Quality Challenges for Inexperienced Measurers", International Workshop on Software Measurement - IWSM 2009 and Mensura 2009, Springer, Amsterdam, The Netherlands, Nov. 2009, pp. 157-169, ISBN 978-3-642-05414-3.

[6] A. Abran, JM Desharnais, A. Lestherhuis, *et al.*, COSMIC-FFP Measurement manual: the COSMIC implementation guide for ISO/IEC 19761:2009, version 3.0.1, Common Software Measurement International Consortium, January 2009.

[7] International Organization for Standardization, ISO/IEC 19761:2011, Software engineering -- COSMIC-FFP -- A functional size measurement method, Switzerland, 2011.

[8] J. Arlow and I. Neustadt, UML 2 and the Unified Process, 2nd edition, Addison-Wesley, 2005.

[9] S. Trudel and J. M. Lavoie, "uObserve Software Specification", v1.0,École de Technologie Supérieure, Montréal (Canada), 2007.

[10] S. Trudel and J. M. Lavoie, "uObserve Software Specification", v2.0, École de Technologie Supérieure, Montréal (Canada), 2010.

[11] A. Abran, Software Metrics and Software Metrology, IEEE Computer Society, Wiley, United States, 2010.