Improving quality of functional requirements by measuring their functional size

Sylvie Trudel[*] *and Alain Abran*[†]

* CRIM/R&D, Montreal, Canada

† École de Technologie Supérieure – Université du Québec/Dept. of Software Engineering and Information Technologies, Montreal, Canada

sylvie.trudel@crim.ca, alain.abran@etsmtl.ca

Abstract:

For many years, the software industry has been applying different types of reviews on their requirements documents to identify and remove defects that would otherwise propagate in the development life cycle, leading to rework and extra cost to fix at later phases. An inspection 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. Requirements documents are also used as input for the measurement of the software size for estimation purposes; when carrying this measurement process, practitioners have often noticed defects in the requirements.

This paper reports on a research project investigating the contribution of the measurers in finding defects in requirements documents. More specifically, this paper describes an experiment where the same requirements document was inspected by a number of inspectors as well as by a number of measurers; the number and types of defects found by both inspectors and measurers are compared and discussed. For this experiment, the measurers used the COSMIC – ISO 19761 to measure the functional size and find defects. Results show significant increase in defects identification when both inspection and functional size measurement are used to find and report defects.

Keywords

Functional requirements, COSMIC, FSM, Functional size measurement, inspection, review.

1 Introduction

Software requirements are written to describe software that will be later developed. Requirements fall usually into two categories: functional requirements and non functional requirements. The functional requirements describe system functionalities while the non functional ones, also called technical requirements and quality requirements, describe required system attributes such as performance, security, and reliability. The focus of the research reported here is on functional requirements.

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 will typically require rework to rectify work done in previous phases of the life cycle.

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-Std-830-1998 [1]: "Correct", "Unambiguous", "Complete", "Consistent", and "Verifiable".

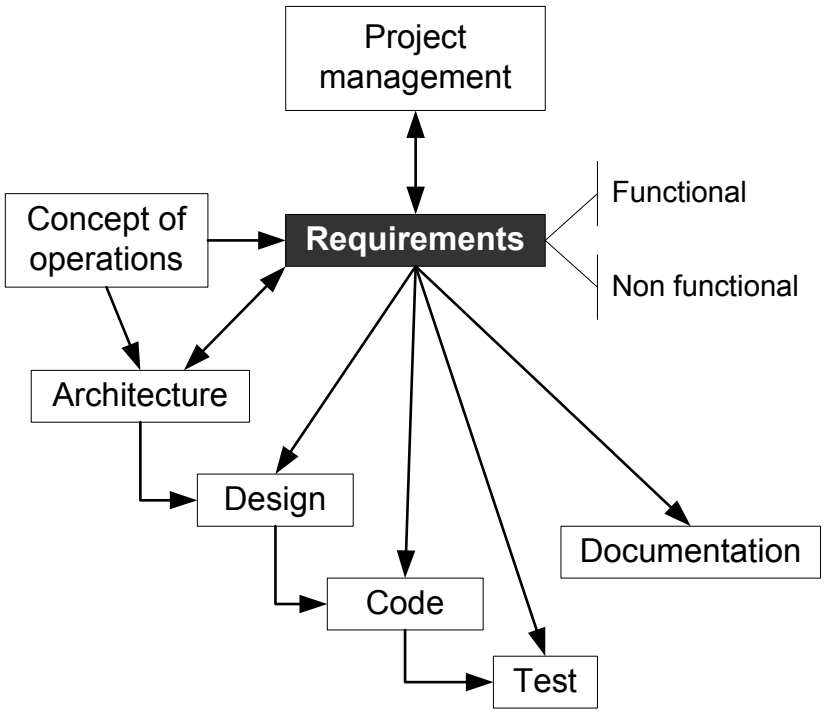

Figure 1: Requirements usage in software development life-cycle phases.

An inspection [2] 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 not been investigated yet and has not been yet documented in the literature as a review technique, even though it is a current measurers practice.

The use of software measurement as a review technique raises a number of questions, such as:

1. Is functional size measurement (FSM) more efficient than inspections for identifying defects in functional requirements?
2. Is functional size measurement (FSM) more effective than inspections for identifying defects in functional requirements?
3. Would it be of value-added to inspections, either for efficiency or effectiveness, if a measurer's role is included?

This paper reports on an experiment carried out to investigate the third question. The experiment reported here was conducted in November 2007 with both industry and academic experts participating to the MENSURA-International Workshop on Software Measurement held in Palma de Majorque (Spain).

For the experiment reported here, the same requirements document was inspected by three inspectors as well as by four measurers. For this experiment, the measurers used the COSMIC – ISO 19761 to measure the functional size and find defects.

1.1 The Inspection Method

The inspection method used in the experiment is an adaptation from Gilb and Graham's work [3]1 . This inspection method contains seven steps as shown in Figure 2.

1 This inspection method has been applied successfully in a Canadian organization more than 2000 times over a four years period and numerous times in other Canadian organizations over the last seven years.

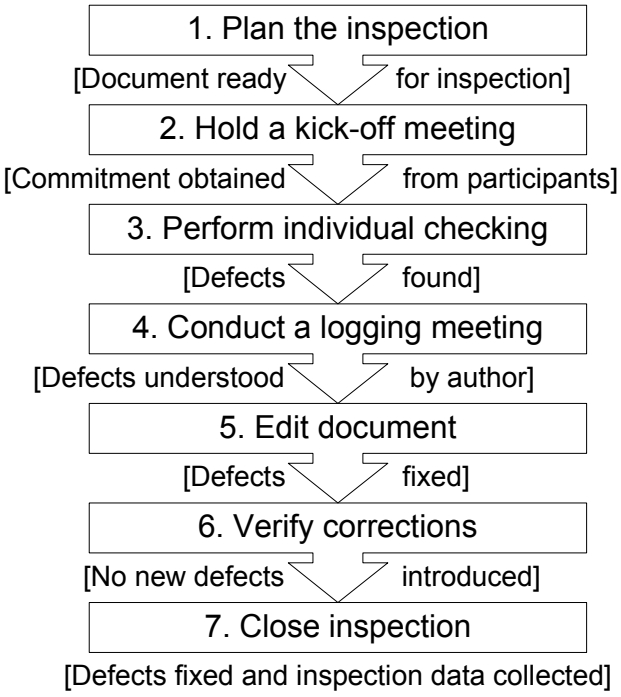

Figure 2: Steps of the inspection method.

1.2 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 functional size measurement method [4] is supported by the Common Software Measurement International Consortium (COSMIC) and is a recognized international standard (ISO 19761 [5]). 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 per-

sistent storage to the functional process. See Figure 3 for an illustration of the generic flow of data groups through software from a functional perspective.

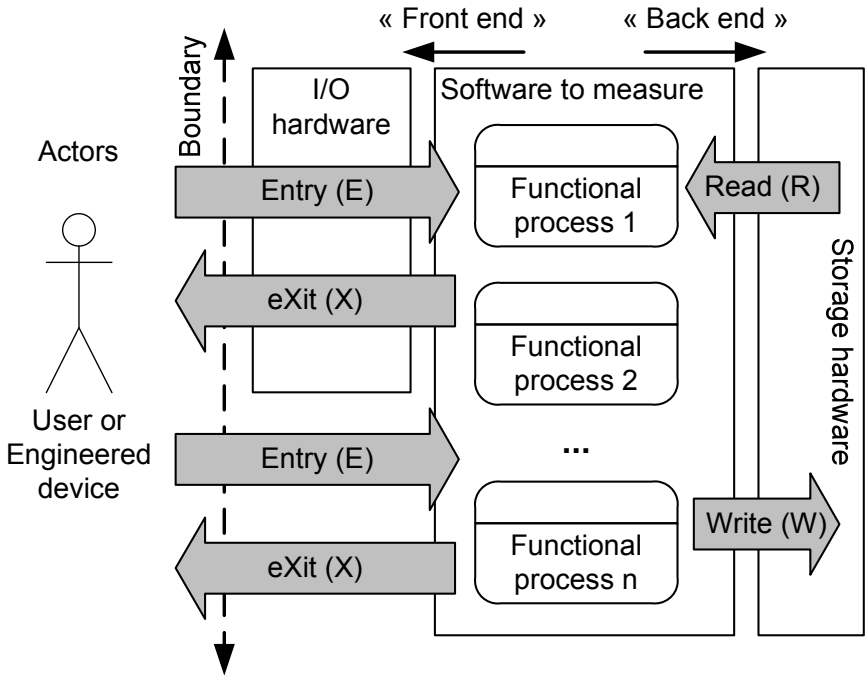

Figure 3: Generic flow of data through software from a functional perspective.

2 The experiment

2.1 Purpose and objective of the experiment

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

The purpose was to perform an experiment involving industry experts, 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.

2.2 The requirements document

The software requirements specification (SRS) document that was chosen for the experiment was compliant with IEEE-Std-830 for its structure and content. This SRS was also compliant with UML 2.0 [6] for the use case diagram, the behavioural state machine, and use case details.

1) SRS overview

The SRS was entitled “uObserve Software Specification” [7] and had 16 pages of descriptive text in English and approximately 2900 words.

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, and assumptions, constraints, and dependencies. Section 3 list 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).

2.3 The participants

1) The inspectors

Three inspectors participated in the experiment. They all cumulate years of industry practice as software practitioners where they had to write and verify software requirements. The first inspector had 8 years of industry practice, she then worked 3 years in a research facility, and she has been teaching software engineering for 4 years during which she participated in industry research projects. The second inspector had over 6 years of industry practice, and has been teaching software engineering for more than 13 years. The third inspector has over 8 years of industry experience and was registered in Ph.D. program in software engineering.

2) The measurers

Four measurers participated in the experiment. They were all COSMIC Certified Entry Level practitioners [8] and were experienced in functional size measurement. All of them were active members of the COSMIC Measurement Practice Committee.

2.4 The experiment steps

The experiment consisted in the following steps applied prior to and during the experiment.

1) Prepare experiment

   a) Prepare material

Prior to the workshop experiment, the chosen SRS was reviewed by a peer to remove most spelling and syntax defects that were injected by the translation of the original requirements document from French to English. Other minor issues were also identified and fixed.

The inspection training material (e.g. templates and procedures) used in this experiment comes from the industry practice of one of the researcher [9].

The experiment material included the chosen SRS, a presentation of the inspection method, the detailed seven steps method, the inspection form for data collection, a defined set of rules, a defined set of inspector roles, definitions for defect and issue types [10] (see Table 1), and definitions for defect categories (see Table 2).

Type	**Definition**
Critical or major	Defect that is likely to cause rework, or prevent understanding or desired functionality.
Minor	Information is wrong or incomplete but does not prevent understanding.
Spelling/Syntax	Spelling or syntax error.
Improvement	The product can stay as is but would be better if the improvement suggestion is implemented.
Question	Any question to the writer of the product.

Table 1: Definitions for defect and issue types.

Improvement suggestions and questions are considered as issues, not as defects. However, a question may later be transformed into a critical or minor defect, depending upon the nature of the question and its related answer.

Category	**Definition**
Functional	Defect related to functional requirements or functional description of the system.
Non functional	Defect not related to functional requirements or to functional description of the system.
Undetermined	Defect that cannot be categorized into Functional or Non functional when first identified.

Table 2: Definitions for defect categories.

Defect categories were defined for analysis purposes, since measurement should primarily be dealing with the functional description of the system to develop.

b) Call for participation

The Call for participation to the experiment was included within the Call for participation to the MENSURA-IWSM-2007, knowing that there was a mix of industry and academic experts. All participants who volunteered for the experiment had previously participated in peer reviews.

2) Provide training on the inspection method

A two-hour training session was provided to all participants on the inspection method, the rules, the roles, and the behaviours to expect and to avoid from inspection participants (inspection leader, author, and inspectors).

3) Perform inspection

a) Plan the inspection

For this experiment, the inspection leader was not given any inspector role: the inspection leader's role was to make sure the process would be followed.

The required roles were chosen from the list of roles (see Table 3). Assigning several inspector roles aims to maximizing defect identification since many perspectives are being applied.

Role	**Definition**
Logic	Focus on logical aspects of the product under inspection, making sure that "everything holds together" (catchall role).
User	Focus on the user or customer point of view (checklist or view point role).
Tester	Focus on test considerations (testability, test requirements, order of testing and order of development for parallel testing, and so on).
Standards	Verify conformity to agreed standards (quality assurance role).

Table 3: Required inspector roles and their definition.

The inspection scope was defined as sections 2 and 3 of the SRS, which size was measured at 2600 words. Thus, planned individual checking effort was set to 1 hour and 45 minutes (105 minutes) based on an inspection rate of 5 pages per hour (one page=300 words). The source documents were the SRS (section 1 – Introduction) itself and applicable standards (IEEE-Std-830 and UML 2.0).

Two inspection modes were defined in the inspection method: “parallel” or “serial”. In “parallel” mode, every inspector has his own copy of the artifact to inspect and they perform their individual checking at the same time. In “serial” mode, only one copy of the artifact to inspect is carried from the first inspector to the last on the inspectors list, allowing inspectors to learn from identified defects by previous inspectors. Because of time constraints of the workshop experiment, the “parallel” inspection mode was applied.

The inspection planning was done prior to the workshop session and required 15 minutes of effort.

b) Hold a kick-off meeting

A brief overview of the SRS was provided to the inspectors. Instructions were given to inspectors to categorize every identified defect into F, N, or U, along with the defect type (see TABLE I).

The Logic role was assigned to inspector #1. The User role was assigned to inspector #2. The Tester and Standards roles were both assigned to inspector #3. All inspectors agreed to play their assigned roles.

From that moment, measurers were asked to leave the room to provide a quiet environment to inspectors.

The inspection kick-off duration was 10 minutes with a total of five participants: three inspectors, one inspection leader, and the writer of the SRS.

c) Perform individual checking

Inspectors performed their individual checking, playing their assigned roles the best they could. Defects and issues were identified and noted on the copy of the SRS of each inspector, along with their respective types and categories. Inspectors stopped the checking activity when they were convinced they had completed the required verification.

Next, each inspector compiled the number of defects per type and reported this data on the inspection form. They also measured their checking effort and compiled it on the inspection form.

d) Perform functional size measurement

The inspection training provided guidance on defect types and categories to measurers, whom attended the session as well. When the writer of the SRS handed a printed copy of the SRS to each measurer, measurers were asked to apply the COSMIC measurement method and to identify any defect and issue, along with its respective type and category.

While inspectors were checking, measurers began the FSM activity, identifying, categorizing, and providing a type for any defect and issue, which may have slowed down measurement.

Each measurer identified functional processes, data groups, and related data movements. Data movements were added to provide the functional size of every functional process. Functional size of each functional process was added to provide the functional size of the system. Once measurers completed the FSM activity, the following data was reported on their inspection forms: effort to measure and identified defects, number of defects per type, and software functional size.

e) Conduct a logging meeting

When both inspectors and measurers had completed their activities, a logging meeting was conducted with the inspection leader, and the inspectors to describe every identified defect and issue. The objective of the logging meeting was for the writer of the SRS to understand all these defects and issues to be able, at the edit phase, to apply an appropriate correction and, if required, a type reclassification (e.g. from Question to Minor or Critical).

The logging meeting duration was one hour (60 minutes), during which all inspectors explained identified defects, focusing on Critical and Minor defect types. The Spelling/syntax type was voluntarily skipped since explanation did not seem relevant. Measurers described only some of their identified defects and the effort it required was negligeable.

At the end of the logging meeting, all SRS hand-written copies were given to the author and experimenter. Later, these copies were scanned individually into a PDF file for verification purposes.

4) Compile experiment data

a) Defects and issues log

Defects and issues were logged on a spreadsheet with the following parameters:

- Location (page #, section #, paragraph #, and line #);
- Description;
- Type (C, M, S, I, or Q);
- Category (F, N, or U);
- Number of inspectors (if more than one identified the same defect or issue);
- Inspectors initials;
- Number of measurers (if more than one identified the same defect or issue);
- Measurers initials;
- Status (Open, Fixed, or Closed); and

- Comment from the researcher.

When appropriate, the researcher reclassified the defect type and category. When two participants identified the same defect with a different type, the defect type that had the most impact was logged (i.e. Critical over Minor).

The spreadsheet allowed filtering data to ease analysis.

b) FSM detailed data

The following FSM detailed data was captured in a spreadsheet:

- Functional process;
- Data groups;
- For each measurer:
  i. Data movements per data group;
  ii. Size per data group;
  iii. Size per functional process;
  iv. System functional size.

c) Effort data

Effort spent per participant for the checking activity and the measuring activity was entered in a spreadsheet. The effort unit of measure was one minute. Effort spent for the other steps of the inspection method was entered separately.

5) Review experiment data with participants

Individual data were isolated and sent to each participant for review and approval. Inspectors reviewed their defects and issues log, and the number of defects and issues per type against the scanned copy of their hand-written commented SRS. Measurers reviewed the same data as inspectors plus their detailed FSM data. Data were hidden from one another to avoid any bias or influence. This step was made to ensure that data analysis would be performed with unbiassed data.

At the time this paper was written, 5 participants out of 7 had sent review feedback with either minor changes or no comment.

6) Analyze experiment data

In industry, FSM is more likely to be performed by a single measurer. Therefore, experimenting with four measurers represents four different experiments.

From the inspection point of view, the industry applies from three to five inspectors for a single inspection of a requirements document. Therefore, data from all three inspectors was combined in a single set of experiment data.

3 The results

3.1 Inspection results

a) Identified defects

The log per participant contained a total of 227 defects and issues, as shown in Table 4.

		Defects			Issues		Total
Type		**C**	**M**	**S**	**Q**	**I**	
Inspectors	Insp #1	20	24	10	1	5	**60**
	Insp #2	10	28	2	0	6	**46**
	Insp #3	7	5	0	0	2	**14**
Measurers	Meas #1	5	1	8	2	1	**17**
	Meas #2	4	2	5	0	0	**11**
	Meas #3	8	14	6	1	0	**29**
	Meas #4	15	11	20	2	2	**50**
Total:		**69**	**85**	**51**	**6**	**16**	**227**

Table 4: Number of defects and issues by type per participant, including duplicates.

Several defects and issues were identified by more than one participant. A total of 191 uniquely identified defects and issues were recorded, as shown in TABLE V, by both inspectors and measurers.

		Defects			Issues		Total
Type		**C**	**M**	**S**	**Q**	**I**	
Category	F	37	55	17	5	4	**118**
	N	21	20	19	1	12	**73**
Total:		**58**	**75**	**36**	**6**	**16**	**191**

Table 5: Number of unique defects and issues by type, by category.

Table 6 shows the 116 uniquely identified defects and issues found by inspectors. Measurers also identified 16 of these 116 defects and issues.

		Defects			Issues		Total
Type		**C**	**M**	**S**	**Q**	**I**	
Category	F	19	39	6	1	3	**68**
	N	17	15	6	0	10	**48**
Total:		**36**	**54**	**12**	**1**	**13**	**116**

Table 6: Number of unique defects and issues by inspectors.

b) Effort spent and effectiveness

Inspectors spent an average of 57 minutes for the checking activity (minimum=55 minutes, maximum=60 minutes). The planned effort per inspector was 105 minutes. Total effort spent by the three inspectors was 170 minutes.

Effort for identifying defects requires not only the checking effort but also effort from previous steps and the logging meeting step [11]. Table 7 provides a summary of effort spent by the inspection team to identify defects.

Inspection step	**Duration**	**# Participants**	**Effort**
Plan the inspection	15 min	1	15 min
Hold a kick-off meeting	10 min	5	50 min
Perform individual checking	--	3	170 min
Conduct a logging meeting	60 min	5	300 min
Total:			**535 min**

Table 7: Effort spent by inspection team.

The effectiveness of an inspection can be calculated as the total effort to identify defects divided by the number of critical defects. In this inspection, the effectiveness is 535 minutes / 36 unique critical defects = 15 minutes per critical defect.

3.2 Measurement results

a) Functional size

Functional size measures in COSMIC Function Point (*cfp*) showed some variations among measurers (see Table 8). Some of these variations in the sizes obtained might be due to defects in the SRS; the sources of these variations will be analyzed in a later phase of this research project.

	Functional size	Average	Standard deviation
Meas #1	62	**59**	**3.3**
Meas #2	55		
Meas #3	61		
Meas #4	57		

Table 8: Functional size per measurer in *cfp*.

b) Identified defects

Measurers have identified between 9 and 39 functional and non functional defects and issues that inspectors did not identify, as shown in Table 9, including duplicates (i.e. defects found by more that one measurer).

		Defects			**Issues**		Total
Type		**C**	**M**	**S**	**Q**	**I**	
Measurers	Meas #1	3	1	5	2	1	**12**
	Meas #2	3	2	4	0	0	**9**
	Meas #3	6	13	4	1	0	**24**
	Meas #4	10	8	17	2	2	**39**

Table 9: Number of defects and issues found by measurers only.

Nevertheless, it was expected that measurers would find a majority of functional defects since the FSM activity focuses on functional description of the software. Table 10 presents the defects found by the measurers when considering only functional defects, including duplicates.

		Defects			**Issues**		**Total**
Type		**C**	**M**	**S**	**Q**	**I**	
Measurers	Meas #1	3	1	4	1	1	**10**
	Meas #2	3	2	3	0	0	**8**
	Meas #3	6	13	3	1	0	**23**
	Meas #4	6	3	6	2	0	**17**

Table 10: Number of functional defects found by measurers only.

Given these figures, what would have been the value-added of individual measurers over the inspection team?

Table 11 provides the number of critical and minor defects, as well as critical only defects, identified by measurers and their relative value-added over the functional defects found by the inspection team.

	Critical & Minor	**Value-added**	**Critical only**	**Value-added**
Inspection team	58	--	19	--
Meas #1	4	7%	3	16%
Meas #2	5	9%	3	16%
Meas #3	19	33%	6	32%
Meas #4	9	16%	6	32%

Table 11: Value added of measurers over inspection team.

All four measurers individually added value to the inspection team efficiency. The increase of defects identification was ranging from 7% to 33% when critical and minor defects are considered. The value-added was even higher when considering only critical defects, ranging from 16% to 32%.

c) Effort spent

Measurers have spent an average of 57 minutes for the measurement activity, including defect identification, as shown in TABLE XII.

	FSM effort	**Average**	**Standard deviation**
Meas #1	49	**57**	**13.4**
Meas #2	45		
Meas #3	60		
Meas #4	75		

Table 12: Effort spent by measurers in minutes.

On average, a measurer took the same amount of effort for performing FSM and identifying defects and issues than an inspector for performing the individual checking step.

In this experiment, the effectiveness of the FSM activity for finding defects cannot be isolated since the effort was spent focusing on sizing the software application.

No time limit was imposed on measurers. However, during the measurement activity, measurers had move to an open space of the conference facility and complained that the noise level had slowed down their measurement pace.

4 Discussion and future work

FSM results typically provides the functional size of the software, allowing a development team or project manager to use this input for estimation and benchmarking purposes. Another important value-added data comes out from this measurement activity is the identification of defects not found by a team of inspectors.

The experiment results demonstrated a value-added on inspection efficiency when having a measurer who raises issues while measuring the functional size. Adding measurement over inspection allowed identifying from 16% to 32% of new critical functional defects, in less effort than the planned individual checking effort. Of course, inspectors do not provide functional size data as it is not part of an inspection method.

Inspectors spent 54% of the planned effort for their individual checking. If the planned checking effort would have been spent totally, inspectors might have found a larger number of defects and issues.

Measurers participating in this experiment may have been over experienced and other less experienced measurers may lead to different results. This will require further experimentation to verify this.

Further work includes other experiments with industry requirements documents that may or may not be compliant with IEEE-Std-830 and UML 2.0.

Acknowledgments

The authors thank participants to the experiment.

The inspectors: Maya Danava, Ph.D., software engineering professor at University of Twente, Netherlands; Mohamad Kassab, test specialist, Oz Communication, www.oz.com, and Ph.D. graduate student, Concordia University,

Canada; and Olga Ormandjieva, Ph.D., software engineering professor at Concordia University, Canada.

The measurers: Harold Van Heeringen, measurement expert, Sogeti, www.sogeti.nl, Netherlands; Luca Santillo, measurement expert, Agile Metrics, www.agilemetrics.it, Italy; Charles Symons, software measurement expert, founder and joint project leader COSMIC, England; and Frank Vogelezang, measurement expert, Sogeti, Netherlands.

References

1. IEEE Computer Society, IEEE-Std-830-1998, IEEE Recommended Practice for Software Requirements SpeciÞcations, New York, NY, June 1998.
2. K. Wiegers, Peer Reviews in Doftware: A Practical Guide, Boston, MA: Addison-Wesley, November 2001.
3. T. Gilb and D. Graham, Software Inspections, Addison-Wesley Professional, December 1993, pp. 13-20.
4. A. Abran, et al, COSMIC-FFP Measurement manual: the COSMIC implementation guide for ISO/IEC 19761:2003, version 2.2, Common Software Measurement International Consortium, January 2003.
5. International Organization for Standardization, ISO/IEC 19761:2003, Software engineering -- COSMIC-FFP -- A functional size measurement method, Frbruary 2003.
6. J. Arlow and I. Neustadt, UML 2 and the Unified Process, 2nd edition, Addison-Wesley, 2005.
7. S. Trudel and J. M. Lavoie, “uObserve Software Specification”, Montreal, Canada: École de Technologie Supérieure, 2007.
8. GÉLOG, “COSMIC Entry Level Practitioners Certificate Holders”, http://www.gelog.etsmtl.ca/cosmic-ffp/entry_level_holders.html .
9. S. Trudel, Software Inspections Workshop, CRIM, Montreal, Canada, 2007.
10. Canadian Department of National Defence, “Defect type definitions”, unpublished.
11. R. Stewart and L. Priven, Revitalizing Software Inspections, presented at the Montreal Software Process Improvement Network (SPIN), Canada, February 6th, 2008.