ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THESIS PRESENTED TO
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

IN PARTIAL FULFILLEMENT OF THE REQUIREMENTS FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
Ph.D.

BY
Sylvie TRUDEL

USING THE COSMIC FUNCTIONAL SIZE MEASUREMENT METHOD (ISO 19761) AS A SOFTWARE REQUIREMENTS IMPROVEMENT MECHANISM

MONTREAL, APRIL 16, 2012

BOARD OF EXAMINERS
THIS THESIS HAS BEEN EVALUATED
BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Alain Abran, Ph.D., Thesis Supervisor
Software Engineering and Information Technology Department at École de technologie supérieure

Mr. Gheorghe Marcel Gabrea, Ph.D, President of the Board of Examiners
Electrical Engineering Department at École de technologie supérieure

Mr. Pierre Bourque, Ph.D, Member of the jury
Software Engineering and Information Technology Department at École de technologie supérieure

Mr. Abdelwahab Hamou-Lhadj, Ph.D., External Evaluator
Electrical and Computer Engineering Department at Concordia University

THIS THESIS WAS PRENSENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND PUBLIC

ON FEBRUARY 24, 2012

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGMENT

First, I would like to thank my thesis director, without whom I would never had started doctorate studies while having a full-time job. You have been a model for me since the day I met you. Your wisdom and encouragements helped me push my limits beyond where I thought they could be.

This research project could not have been done without the 35 participants who volunteered in the experiments that took place. Special thanks to the five expert measurers who were very generous of their time and support: Jean-Marc Desharnais, Harold Van Heeringen, Luca Santillo, Charles Symons, and Frank Voglezang.

I would like to thank my employer, Pyxis Technologies, and more specifically François Beauregard, Martin Proulx, Maurice Bergeron, Marion Gilbert, Tremeur Balbous, Yves Ferland, and all my colleagues who have generously accepted the time-off I required to complete this thesis. I am very grateful to Jean-Marc Lavoie, co-author of the uObserve SRS document, who generously agreed to the usage and distribution of this document throughout my research project. I am also grateful to Pierre Bergé, my writing process coach, for his immense talent and generosity. A special thank to Mathieu Boisvert, my co-author of the book 'Choisir l'agilité', for his understanding when I put pressure on the book writing schedule so that I could concentrate on this thesis sooner than later.

Finally, I would like to thank Yves, my partner in life, for his understanding and great support: you are my best friend and I love you.

UTILISATION DE LA MÉTHODE DE MESURE DE LA TAILLE FONCTIONNELLE COSMIC (ISO 19761) COMME MÉCANISME D'AMÉLIORATION DE LA QUALITÉ DES EXIGENCES FONCTIONNELLES DES LOGICIELS

Sylvie TRUDEL

RÉSUMÉ

Ce projet de recherche porte sur la contribution de l'utilisation de la méthode COSMIC (ISO 19761) de mesure de la taille fonctionnelle du logiciel à l'identification des défauts dans les spécifications fonctionnelles. Le mesureur de la taille fonctionnelle doit comprendre et interpréter les exigences fonctionnelles servant d'intrant à un mesurage. Dans l'industrie ces exigences sont typiquement écrites en langage naturel et sont propices aux ambigüités: des erreurs d'interprétation peuvent donc en découler. Les erreurs dans les spécifications fonctionnelles affectent le coût des étapes subséquentes du cycle de développement et d'évolution du logiciel: du travail additionnel est requis lorsque ces erreurs sont détectées et doivent être corrigées. Les organisations appliquent alors des techniques de revue et d'inspection permettant de détecter les erreurs dans les spécifications. Toutefois, ces techniques ne permettent pas de détecter l'ensemble des défauts dans un seul cycle de revue ou d'inspection, laissant ainsi un certain nombre de défauts résiduels dans le document de spécifications. Nonobstant l'application de telles techniques, un mesureur appliquant la méthode COSMIC peut identifier des défauts dans les spécifications fonctionnelles lorsque celles-ci ne définissent pas clairement les éléments attendus en intrant au mesurage.

Les objectifs de ce projet de recherche sont de quantifier l'efficience et l'efficacité (coût unitaire) de l'utilisation de la méthode COSMIC en tant que mécanisme d'identification des défauts dans ces spécifications fonctionnelles, en comparant les résultats avec une approche d'inspection. L'efficacité et le coût unitaire sont établis à partir du nombre de défauts identifiés et de l'effort pour y parvenir.

Les résultats démontrent que, en moyenne, les mesureurs participant à une inspection trouvent un nombre de défauts similaire à celui obtenu par l'ajout d'un inspecteur, ce qui maintient l'efficacité. En moyenne, le coût unitaire augmente légèrement puisque l'effort est légèrement supérieur, sauf qu'en plus, à ce coût, le projet obtient une mesure de la taille fonctionnelle utilisable à des fins d'estimation, d'étalonnage et de supervision des améliorations de processus. Sans l'apport d'identification des défauts, le mesurage de la taille fonctionnelle est considéré comme un coût de gestion. Avec l'identification des défauts de façon similaire à un inspecteur, le mesurage de la taille fonctionnelle peut alors être considéré comme un bénéfice qui crée de la valeur liée à l'économie de travail à refaire, puisque les défauts identifiés par un mesureur peuvent être corrigés relativement tôt dans le cycle de développement.

À partir des expérimentations avec des experts et des praticiens dont l'expérience était limitée, il a été de plus observé que les praticiens nouvellement formés à la mesure de la

taille fonctionnelle rencontraient plusieurs défis liés à la qualité de leurs résultats de mesure. Ce projet de recherche a donc contribué aussi à définir des exigences liées à la formation des mesureurs.

THE COSMIC ISO 19761 FUNCTIONAL SIZE MEASUREMENT METHOD AS A SOFTWARE REQUIREMENTS IMPROVEMENT MECHANISM

SYLVIE TRUDEL

ABSTRACT

This research project investigates the contribution of the COSMIC software functional size measurement method (ISO 19761) to identify defects in functional requirements. The functional size measurer has to understand and interpret the functional requirements used in c input to the measurement process. Industry requirements are typically written in natural language and are prone to ambiguities: therefore, interpretation errors do happen. Errors in functional requirements affect the cost of subsequent steps of the software life cycle through rework when these errors are detected and corrected. Organizations then apply review and inspection techniques to detect errors in requirements. However, these techniques do not detect all defects in a single cycle of review or inspection, leaving a number of residual defects in the requirements document. Notwithstanding the application of such techniques, a measurer using the COSMIC method can identify defects in the functional requirements that do not clearly define the elements required as input to the measurement activity.

The objectives of this research project are to quantify the efficiency and effectiveness (in terms of unit cost) when using the COSMIC method as a technique for identifying defects in these functional specifications, comparing the results with an inspection method. The efficiency and unit costs are computed using the number of defects identified and the effort to do so.

Results show that, on average, a measurer participating in an inspection finds a number of defects similar to the number of defects found by the addition of an inspector, maintaining the efficiency. The average unit cost increases slightly as the effort is slightly higher, except that for this cost, the project gets the software functional size usable for estimation, benchmarking, and process improvement monitoring. Without the identification of defects, the functional size measurement is considered as a management cost. With the additional benefit of the identification of defects, the functional size measurement creates added value in terms of rework costs savings since the defects identified by a measurer can be corrected earlier in the development cycle.

In the experiments with experts and practitioners whose experience was limited, it was also observed that the practitioners newly trained to functional size measurement faced several challenges which impacted the quality of their measurement results. Therefore, this research project also contributed in defining requirements for the training of measurers.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABREVIATIONS

Appx#	Appendix number.
CMMI	Capability Maturity Model Integration.
ConOps	Concept of Operations.
COSMIC	Common Software Measurement International Consortium.
CRIM	Centre de Recherche Informatique de Montréal.
DM	Data movement.
DoD	Department of Defence.
E	Entry [data movement].
ÉTS	École de Technologie Supérieure.
FFP	Full Function Point.
FiSMA	Finnish Software Metrics Association.
FP	Function Point.
FPA	Function Point Analysis.
IEC	International Electrotechnics Commission.
IEEE	Institute of Electrical and Electronics Engineers.
IFPUG	International Function Point Users Group.
IJCIS	International Journal of Computer & Information Science.
ISO	International Organization for Standardization.
IWSM	International Workshop on Software Measurement.
LESIA	Laboratoire d'environnements de synthèse et interfaces avancées.
MENSURA	Conference on Software Process and Product Measurement.
Mil	Military.

MPC	Measurement Practice Committee.
NESMA	Netherlands Software Metrics Users Association.
QA	Quality Assurance.
R	Read [data movement].
SERA	Software Engineering Research, Management & Applications.
SMO	Small and Medium Organizations.
S.O.N.I.A.	Système d'opération nautique intelligent et autonome.
SSR	Software Specification Review.
Std	Standard.
SWEBOK	Software Engineering Body of Knowledge.
W	Write [data movement].
X	eXit [data movement].
XML	eXtended Markup Language.

LIST OF SYMBOLS

CFP	COSMIC Function Point.
Cfsu	COSMIC functional size unit.
hrs	Hours.

INTRODUCTION

This thesis contains seven chapters and 15 appendixes. This introduction outlines the thesis organization.

Chapter 1 presents the literature review on techniques to improve the quality of requirements as well as on the use of functional sizing techniques to improve the quality of requirements. . Chapter 2 presents the research issues and research objectives. It also presents the research motivation and an overview of the research methodology depicted in four phases.

Chapter 3 presents the pilot project, its results, and lessons learned leading to experimental material requirements and selection of approaches and methods.

Chapter 4 presents the experiments conducted during Phase 2 of the research project. These experiments aimed at quantifying the effectiveness and efficiency of using the COSMIC method as a means to identify defects in software functional requirements, and comparing the results with an inspection approach. Experiments were conducted with experts as well as with practitioners with limited experience. Data was gathered, verified, and analyzed regarding defects and issues found, effort spent, and functional size. This chapter 4 concentrates on defect and effort data and quantification of effectiveness and efficiency.

Chapter 5 explores the relationship of defects and functional size. It begins with detailed functional size data gathered during Phase 2 and analysis thereof. This chapter describes Phase 3 of the research that consists in updating the experiment material based on defects identified by experiment participants. Then, in Phase 4, the updated functional requirements were measured again by the same experts and by a subset of practitioners with limited experience in order to observe the impact of defects on functional size.

Chapter 6 discusses the results and potential solutions to observed challenges or problems.

Chapter 7 contains a list of recommendations based on the research findings.

CHAPTER 1

FUNCTIONAL REQUIREMENTS QUALITY AND SIZING IN THE LITERATURE

1.1 Functional requirements

1.1.1 Requirements usage in the software development life-cycle

The requirements describe the software system to be developed. They are generally divided into two categories: functional requirements and non-functional requirements. The first category describes the system functions while the latter, also known as technical and quality requirements, describes the desired system attributes such as performance, safety and reliability. This research focuses only on software functional requirements. Requirements are the basis of the whole development process since they are usually in input to all phases of the development cycle, as shown in Figure 1.1.

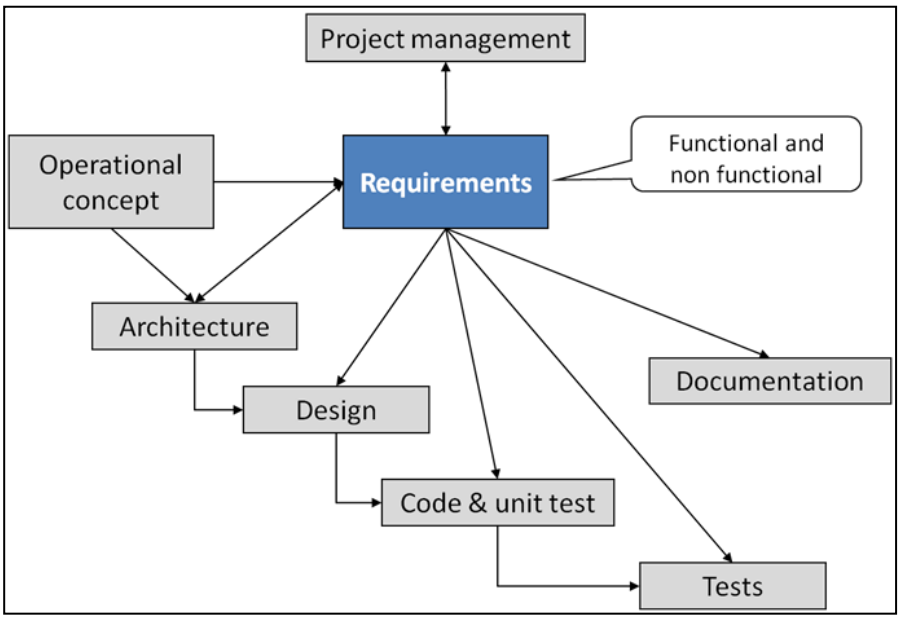

Figure 1.1 Software requirements usage in typical phases of a software development cycle.

In order to document the reasons for a project to develop software, an operational concept document will be created, sometimes referred to a 'vision document' or 'principles and guidelines' of the system according to the adopted methodology, and will contain a list of the major system functions (high-level requirements) and scenarios of operations. When the development project officially starts, the requirements documentation is created. Notwithstanding the selected software development life cycle (e.g. waterfall, iterative, incremental, or prototyping, to name a few (Pressman, 2001)), the requirements are required and their elicitation mechanisms are numerous (Leffingwell and Widrig, 2003).

1.1.2 Elicitation mechanisms

Functional requirements describe what a system has to do, from all of its users' viewpoints. Chapter 2 of the SWEBOK Guide (Sawyer and Kotonya, 2002) (Abran *et al.*, 2002) describes several mechanisms for documenting requirements. Other authors (Leffingwell and Widrig, 2003; Sommerville, 2004; Thayer and Dorfman, 2000; Weigers, 2003) also describe a variety of mechanisms or forms that functional requirements can take, including:

- Natural Language (Weigers, 2003).
- Use cases (Jacobson, Booch and Rumbaugh, 1999).
- User stories (Beck, 2000).
- Graphical user interface and report prototypes (Ducharme and Trudel, 2004).
- State diagrams (also called 'finite state machines') (Leffingwell and Widrig, 2003).
- Entity-relationship diagrams (Chen, 1977).

These different mechanisms for writing requirements are not mutually exclusive and can be combined when necessary. These requirements could be merged either into a single document or into a series of documents, whichever is most appropriate for the project. Often it may be appropriate to adopt a documentation standard, which could have been defined by the organization, or acquired in combination with a development methodology or it could refer to an industry documentation standard that is generally available in the market.

1.1.3 Documentation standards

Several organizations have developed standards for documenting software requirements in order to structure the information found therein. Documentation standards generally indicate an ordering, section titles and the type of information they should contain. These standards specify the 'what' without providing any indication on the 'how'. They do not refer to a methodology or an elicitation technique: the choice is left to a decision of the software project.

Among these standards, there are:

- IEEE Std 1362-1998 IEEE Guide for Information Technology System Definition - Concept of Operations (ConOps) Document (IEEE, 1998a).
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications (SRS) (IEEE, 1998b).

1.1.3.1 IEEE Std 1362-1998

The ConOps in IEEE Std 1362 (IEEE, 1998a) aims to document the operational concept envisioned for the software-intensive system. Without describing in detail the system, its content establishes a relationship between the users' needs and vision and the considered technical solutions. In summary, the operational concept standard includes the following sections:

- Section 1: Introduction, context, and scope.
- Section 2: Referenced documents.
- Section 3: Description of the current situation or the current system to change.
- Section 4: Purpose and nature of proposed changes (problem).
- Section 5: Concepts of the proposed system.
- Section 6: Operational scenarios which outline how the proposed system will be used by different types of users.
- Section 7: Operational impacts, organizational impacts, and impacts during development.

- Section 8: Analysis of proposed system improvements, including a summary of the advantages and disadvantages and alternatives and compromises that have been considered.

When reading a ConOps document, the reader should obtain an overview of the current problem and an understanding of the proposed solution. This type of document is often used as a basis for the decision to go ahead with the development project.

1.1.3.2 IEEE Std 830-1998

The IEEE Std-830 (IEEE, 1998b) software requirements specification (SRS) document aims to detail the functional, technical and quality requirements of the software to develop. In summary, the SRS standard contains the following sections:

- Section 1: Introduction, purpose and scope.
- Section 2: overview of the system and its interfaces.
- Section 3: Specific requirements (their ordering can take many forms, some of which are proposed in Section 5.3.7 of this standard) such as:
  - External interfaces.
  - Functionalities.
  - Performance requirements.
  - Data and database requirements.
  - Design constraints.
  - Compliance with standards.
  - The software system quality attributes.
  - Any other requirements.

An SRS document should be read and understood by both users and developers to reduce the risk of developing a software that does not comply with users needs. In this context, it becomes important to focus on specific quality attributes of the content and then apply verification practices to achieve them.

1.1.4 Quality attributes

Section 4.3 of IEEE Std 830 provides a list of desired quality attributes of a software specification, which are:

> "An SRS should be:
> a) Correct;
> b) Unambiguous;
> c) Complete;
> d) Consistent;
> e) Ranked for importance and/or stability;
> f) Verifiable;
> g) Modifiable;
> h) Traceable."

The attributes descriptions in IEEE Std 830 explain why they should be achieved without specifying what must be done to achieve these attributes.

In this research project, the selected focus is on the first four attributes: correct, unambiguous, complete, and consistent.

1.1.5 Software engineering best practices models

Several standards and best practices models are available to software development organizations to guide them in developing their capabilities to develop and deliver software within budget and schedule while being consistent with users stated and expected needs. Among these models, there is the Capability Maturity Model Integration (CMMI) (Chrissis, Konrad and Schrum, 2003) from the Software Engineering Institute (SEI), funded by the U.S. Department of Defense (DoD). This model also includes practices of several other disciplines such as systems engineering, subcontracting, and teamwork. The CMMI describes 'what' the process should do without specifying 'how' it should be done.

The CMMI includes 22 process areas (PA), each containing a number of specific and generic practices grouped under objectives. Two of the process areas are directly related to

requirements: ‘Requirements Development’ and ‘Requirements Management’. In the first PA, the objectives are to develop customer and product requirements as well as to analyze and validate them. In the second PA, the sole objective is to manage requirements throughout the project, specifically changes to requirements that will occur inevitably during the project. A third PA in the CMMI is also relevant to this research, ‘Verification’ which includes, among other things, peer reviews and related specific practices.

1.1.6 Practices leading to requirements quality

There exist several requirements elicitation mechanisms (Leffingwell and Widrig, 2003). However, once written by its authors, there is in practice no guarantee that the requirements are consistent with the desired quality attributes (Pressman, 2001). When attempting to achieve these quality attributes, organizations use a set of practices such as peer reviews and formal requirements reviews (Kotonya and Sommerville, 1998). Different approaches and techniques for implementing these practices are discussed in the next section.

1.2 Ensuring quality of requirements

1.2.1 Reviews applied to requirements

Standards and technical reports covering the software development process as a whole have been published by the Institute of Electrical and Electronics Engineers (IEEE) and by the International Organization for Standardization (ISO). These standards and technical reports include review activities of several project artefacts, including requirements. The DoD has published two of these standards in the ‘90s: DOD-STD-2167A (DoD, 1994) and Mil-Std-498 (DoD, 1998) documents. Both of these standards describe a formal software requirements review activity called ‘Software Specification Review (SSR)’ and in which several persons play the following roles: project manager, engineer(s) who wrote the software specifications, a customer representative and a responsible for quality assurance. All these participants should have received the requirements document at least one week prior to

the review meeting where they discuss issues to be resolved with regards to the required quality attributes. Minutes of the review meeting contain each issue to be addressed in order to follow up on modifications. Upon participants' request, a second requirements review meeting could be held when the corrected version of the requirements becomes available.

The DoD standard was later replaced by an international standard for commercial use, ISO 12207 (ISO, 2008), which also describes a formal requirements review activity.

In some organizations, ad hoc readings of requirements documents can be made by users and developers, often in isolation. However, this less formal form of review does not provide all the benefits of formal reviews (including peer reviews discussed in the next section) since there are no rules ensuring the quality of the review.

1.2.2 Inspections and peer reviews approaches

A peer review approach was first published in 1976 by Fagan (Fagan, 1976) who named this approach 'inspection'. Several inspection approaches have since been published and are discussed in the next sections.

Inspections were initially applied to software design and code since the analytical models describing, among other things, the functional specifications were not widespread at that time. Structured software development methodologies appeared soon after, such as (Chen, 1977), (Yourdon and Constantine, 1978), (DeMarco, 1979), and (Gane and Sarson, 1982), and in which one or more analysis models were formalized.

From the time when organizations began documenting their analysis models, it became possible to improve their quality by applying a review approach. But, like any other review approach, an inspection does not guarantee that all defects will be found. With the publication of the Capability Maturity Models (CMM (Paulk *et al.*, 1993) and CMMI (Chrissis, Konrad and Schrum, 2003)) in the late 1980s and 1990s, organizations began to

apply more and more peer reviews in addition to formal reviews at development project milestones. The usage of several types of reviews aims at increasing the review efficiency, which is defined as the ability of a software team to identify defects in an artefact and remove them (Gilb and Graham, 1993).

All published peer review approaches apply the same concepts and have essentially the same steps, although naming differs from one another – see Table 1.1. In general, inspection approaches define a series of seven steps, participant roles, and inputs and outputs.

Each step can be summarized as follows:

1) Planning: this step aims to coordinate inspection participants, select appropriate participants, and manage the schedule.
2) Overview: this step is to explain the context of the document to be inspected and the inspection objectives so that each inspector understands how the inspection preparation should be made.
3) Preparation: for each inspector, this step is to identify the greatest number of potential defects with the help of checklists and rules. To be effective, each inspector must spend a minimum of effort, usually proportional to the number of pages to be inspected.
4) Meeting: this step aims to share knowledge on potential defects, raising questions for the author to respond, and note any concern. According to the method used, a person will record each item that the author will address later. The moderator ensures that the method is followed, verify that the participants do not attempt to solve every identified defect, and monitors the participants' behaviour so that the meeting takes place in a professional and friendly atmosphere and that all are actively involved.
5) Rework: during this step, the author of the requirements document corrects the document in accordance with potential defects, questions and concerns that were raised and recorded in the meeting.
6) Follow-up: this step is to ensure that each identified defect has been corrected as appropriate. A participant, who may be the moderator or one of the inspectors, verifies

between the initial version of the document and the corrected version that recorded issues were adequately corrected and that no new defect was inserted.

7) Causal analysis: this step aims to identify root causes of defects among the most important ones. It is usually done when there is a documented organizational process and when people are dedicated at improving these processes.

Table 1.1 Naming similarities and differences among 3 peer review approaches

Approach element	Generic approach	Fagan	Gilb & Graham	CRIM
Process steps	Planning Overview Preparation Meeting Rework Follow-up Causal analysis	Planning Overview Preparation Meeting Rework Follow-up Causal analysis	Planning Kick-off Checking Logging meeting Editing Follow-up Process brainstorming	Plan the inspection Open the inspection Inspect the product Explain issues Update the product Re-check / Close Process improvement[1]
Roles	Author Moderator Inspectors Reader Recorder	Author Moderator Inspectors Reader Recorder	Author Inspection leader Inspectors (n/a) Scribe	Author/editor Inspection leader Inspectors (n/a) (n/a)
Inspection team size	Variable	Always 4	2 to 5, as required by product type	Same as Gilb & Graham
Particul-arity	(n/a)	During the meeting, the reader paraphrases the content of the artefact	No reading. Checking is done against source documents, relevant sets of rules, and checklists	No scribe or recorder. Defects are noted by each inspector on the product copy. Plus, same particularity as Gilb & Graham.
Defect type or severity	Major Minor Spelling	Major Minor	Major and Fatal[2] Minor Spelling	Critical Minor Spelling/syntax

Note 1: This step is an optional activity at the end of the "Explain issues" step.
Note 2: Gilb & Graham propose this naming with provisions for other naming conventions as deemed appropriate by the organization adopting their approach.

The following sub-sections briefly describe characteristics of a number of peer review approaches when applied to functional requirements.

1.2.2.1 The Fagan approach

In the Fagan inspection approach (Fagan, 1976), all inspectors work on their own copy of the requirements document, at about the same time, during the preparation step. There are always four inspectors per inspection. During the meeting, a reader paraphrases the functional requirements. This approach allows inspectors to see if they understood the meaning of the requirements during their preparation. A 'recorder' notes each potential defect as identified by the inspectors.

1.2.2.2 The Gilb and Graham approach

The terminology of the Gilb and Graham approach (Gilb and Graham, 1993) differs from the Fagan approach. The overview is named 'kick-off meeting', the preparation is named 'individual checking', the meeting is called 'logging meeting', the correction is called 'editing', the causal analysis is named 'process brainstorming', and the moderator is named 'inspection leader'. Depending on the author objectives about the deliverable to be inspected and the type of this deliverable, two to five inspectors can be selected. The inspection leader assigns distinct inspection roles to the participants so that the deliverable is inspected from different perspectives.

During preparation, inspectors will systematically compare each sentence or paragraph of the inspected document with any source document, rules, and checklist items. All rules have a unique identifier and each identified defect must refer to a rule to ensure objectivity. The Gilb & Graham approach advocates a rather slow pace of preparation, such as one page per hour, a page being equivalent to 300 words. This slow preparation pace allows identifying as many defects as possible. The inspected document is printed with page numbers and line numbers. Each inspection scope is limited to a number of pages equivalent to two hours of preparation, because inspectors can hardly stay focused beyond this duration. Therefore, a

large requirements document is likely to be divided into ‘chunks’ of 3000 words or less, allowing inspections on portions considered done before the whole document is completed.

During the meeting, defects are recorded by the scribe on a separate list, carefully identifying the page number, the line number, the violated rule, the issue severity, and a brief description. This defects and issues list will be used for the causal analysis step later on.

Emphasis is put on quantitative analysis, as more than 30 measures are collected throughout an inspection. The inspection approach authors consider those measurement results and their analysis as critical to maximize the inspections performance, as well as defect prevention.

Six forms are used to manage an inspection:

1) A request for inspection.
2) An inspection master plan.
3) A summary of inspection, which contains cumulative data from a single inspection.
4) A confirmation of completion.
5) A list of identified potential defects.
6) A notice of defect identification when inspectors detect at least one potential defect in a source document.

1.2.2.3 The CRIM adaptations to the Gilb & Graham approach

The CRIM inspection approach (Trudel, 2007) is an adaptation of the Gilb & Graham approach. After having used from 1996 to 2000 the Gilb and Graham inspection approach at a defence contractor – which software process capability was evaluated twice at level 2 of the CMM during that period – the inspection practitioners (including this thesis author) adapted and improved the approach in order to reduce the overhead cost of using the first four forms, combining them into a single form. Moreover, instead of having a separate list of defects, these were recorded directly on the document using a coding convention for severity and follow-up. The outcome was an adapted inspection approach that was adopted by the Centre

de Recherche Informatique de Montréal (CRIM), a Montreal research center which made this new approach available to the industry upon request. This CRIM approach (Trudel, 2002) was also adapted for small organizations.

Two inspections modes are available in the CRIM approach:

1) The ‘parallel’ mode in which each inspector has his own copy of the product and inspects approximately at the same time as the others.
2) The ‘serial’ mode in which a single copy of the inspected document passes from one inspector to another.

The ‘serial’ mode has the advantage of accelerating knowledge transfer between inspectors, eliminating the difficulty of accurately measuring the number of defects (because there are defect duplicates in almost every inspection in the ‘parallel’ mode), and providing the author a single working copy to track corrections.

1.2.2.4 Other peer review approaches

In his book on peer reviews (Weigers, 2002), Wiegers identifies two other approaches, in addition to those of Fagan and Gilb & Graham. The first is the ‘High-Impact Inspection’ approach, published by David Gelperin and his colleagues, focusing on four to six critical aspects of the artefact to be reviewed. Wiegers also mentions the ‘Phased Inspections’ approach, by John Knight and Ann Myers, focusing on one property of the artefact to be reviewed, such as portability, maintainability or completeness. Inspectors must inspect the same artefact for each property to be verified.

1.2.3 Benefits of applying peer reviews to requirements

Organizations which implement peer reviews may measure or observe benefits such as reduced testing effort, reduced maintenance effort, increased process efficiency (i.e. fewer defects found by customers), reduced rework, and improved knowledge sharing among team

members. However, the benefits may be null when inspections are poorly executed and inspectors do not find defects or when identified defects are not corrected.

1.2.4 Peer review measures

To fulfill various information needs about the software development process, more specifically related to processes aiming at requirements quality, organizations are using multiple measures. Managers can take several decisions based on these measurement results.

1.2.4.1 Base measures

Organisations implementing peer reviews, or other forms of requirements review, usually measure the following:

- Number of defects found and corrected: the number of defects may be collected during inspections, during testing, and post-deployment for a given duration.
- Review effort: the effort taken for a review may be collected at each inspection step, for every inspection, and for all inspections in a given period.
- Size of the reviewed artefact: the size of the reviewed artefact can be physical such as the number of pages or number of Source Lines of Code (SLOC), or this size can be of the functional requirements measured using an international standard, such as IFPUG function points (ISO, 2009) or COSMIC function points (ISO, 2011b).

When taken in isolation, these measurement results may not lead managers to take specific decisions. It is by combining them together to compose derived measures and by observing their trends over time that managers can make informed decisions related to the software process (Gilb and Graham, 1993; Weigers, 2002).

1.2.4.2 Derived measures

This section describes some measures that organizations derived from base measures. For each of these derived measures, their composition is described and an example of a decision that could be made is provided.

- Effectiveness = review effort / number of defects found and corrected during inspection (in hours per defect). The effectiveness derived measure is equivalent to a unit cost derived measure as it is expressed in effort per defect. Organizations can compare their inspection effectiveness with industry data or more specifically between the different artefacts from their software development process. Thus, when the effectiveness value is above a predetermined threshold for a given artefact (e.g. the code inspection effectiveness), managers may decide not to apply inspections to this artefact or to apply them only partially (e.g. 20% of the most critical or complex components).
- Efficiency = number of defects found and corrected during inspections / total number of defects (as a percentage). This derived measure gives the percentage of defects found in inspections. Managers looking at this derived measure often set a goal of quality for their software development process (e.g. 97% of defects must be found before delivery to customers, and over 75% of these defects must be found through inspections) . When the actual value of efficiency is too low, managers can assign improvement actions aiming to increase inspections efficiency (e.g. improving the use of checklists, better trained inspectors, strengthening the usage of inspector roles, whichever is appropriate).
- Defect density = number of defects / size (e.g. 25 defects per 1000 SLOC or two defects per 100 function point). Organizations monitoring the improvement of their software development processes are interested to monitor the defects density trend over time (e.g. every 3 to 6 months) to take action when the defect density does not drop in accordance with preset quality objectives.
- Correction density = correction effort / size (in hours per size unit). This derived measure is used to compare a normalized correction effort by size through various projects. Managers are concerned with projects for which the correction effort is too high

compared to other projects in order to take action to avoid budget or schedule slippage in the subsequent project phases.

1.3 Measuring software functional size

Relatively early in the history of computing, organizations were concerned about development costs and duration (Boehm, 1981). Researchers then tried to estimate these costs and durations from various project parameters, using information from past projects such as the technology, the programming language, the number of programmers in the team and their relevant experience, and the size of the software systems that had been developed. This size was mainly used in estimation and productivity models (dollars or hours per size unit) and to compare software systems.

The previous section indicated that the size is a base measure that is used in several peer reviews derived measures. Software project cost and duration measures can also be added, both estimated and actual. This software size was expressed in different units of measurement such as lines of code and function points. The first unit of measurement refers to a physical size while the latter refers to a functional size. These two concepts are further described in the following sections.

1.3.1 Physical software size with lines of code – SLOC

During the 1960s, some organizations began to measure the software size in SLOC. To estimate the cost of a new project, some organizations estimated the number of lines of code the software would have, and then applied their estimation model. The use of size in SLOC made sense for software systems written with the same programming language. This measure had to be taken when the system was completed and operational, so relatively late in the development cycle.

Using the estimated or measured size in lines of code resulted in a number of concerns (Jones, 1996), including:

1) Difficulty of using size measures from previous projects if the software was developed in a programming language different than the one about to be developed.
2) Difficulty or inability to obtain a homogeneous measurement result when several programming languages are used throughout the same project.
3) Often important differences between the estimated size and the measured size at the end of the project.
4) Size known only at the end of the project.
5) Difficulty in defining what 'one line of code' is, whether the comment lines are included or not, if each of the physical lines are counted or only the statements are counted, even when they spread over several physical lines.

1.3.2 Functional size with function points

Functional size measurement (FSM) is a means for measuring the size of the software requirements of an application, regardless of the technology used to implement it. FSM was introduced by Allan Albrecht of IBM in 1979 (Lokan, 2004) with a method called Function Point Analysis (FPA). At that time, Albrecht aimed to obtain a size measure for use in a productivity model (effort per size unit). Since its first publication in 1979, the FPA method has evolved and several other methods have been developed, mainly inspired by Albrecht's work. These methods are briefly described in the following sub-sections.

1.3.2.1 Albrecht FPA

This FPA size measure was intended to represent the size of the functionality delivered to users, independent of the technology used to develop the software. Emphasis was placed on visible aspects to users as the inputs (data input), outputs (data display or printed), requests, and master files. Unlike lines of code, measuring FPA was obtained earlier in the software

development life cycle. By their nature, the original design of the function points applies primarily to the information systems domain.

A second version of Albrecht measurement method was published in 1984. The measure was named ‘function point’ for the first time.

1.3.2.2 IFPUG

Two years later, users of the FPA method created a not-for-profit organization named ‘International Function Point User Group’ (IFPUG). This group aims to publicize functions points but also to standardize the function points counting practices and usage. For this, IFPUG publishes guides, measurement manuals, and ‘white papers’. They organize conferences and hold certification exams for individuals wishing to obtain the title of ‘Certified Function Point Specialist’. IFPUG has also evolved the method over the years: version 3.0 in 1990, 4.0 in 1994, 4.1 in 1999, 4.2 in 2004, and latest standard release in 2009 (ISO, 2009).

1.3.2.3 Mark II

The Mark II method (ISO, 2002c) was designed by C. Symons to address some weaknesses of the IFPUG method. The author presented the Mark II method as an evolution of the IFPUG method. Mark II is mainly used in the United Kingdom.

1.3.2.4 NESMA

The Netherlands Software Metrics Users Association (NESMA) method (ISO, 2005) is an adaptation of the IFPUG method published by the users association for software measurement of the Netherlands.

1.3.2.5 FiSMA

The Finnish Software Metrics Association (FiSMA) method was published as an international standard by the software measurement association of Finland in 2010 (ISO, 2010). The FiSMA method is derived from the IFPUG method.

1.3.2.6 COSMIC

The scope of the IFPUG method being limited to information systems, Abran and his colleagues first proposed an extension of the IFPUG method in 1997 in order to also measure embedded and real time applications and their software measurement method was initially called ‘Full Function Points’ (FFP). They subsequently created a group in 1998 called ‘Common Software Measurement International Consortium’ (COSMIC). Modifications were brought to the COSMIC method to simplify and broaden its usage from a metrology perspective and its version 2.2 was published as an international standard (ISO, 2003b). The method continued to evolve and version 3.0 was published in 2007 (Abran *et al.*, 2007).

1.3.3 Overview of software size history

Although other software sizing methods have been published over the last 20 years, only those listed above have been accepted by the international community as ISO standards. Figure 1.2 graphically presents a publication history summary of these methods.

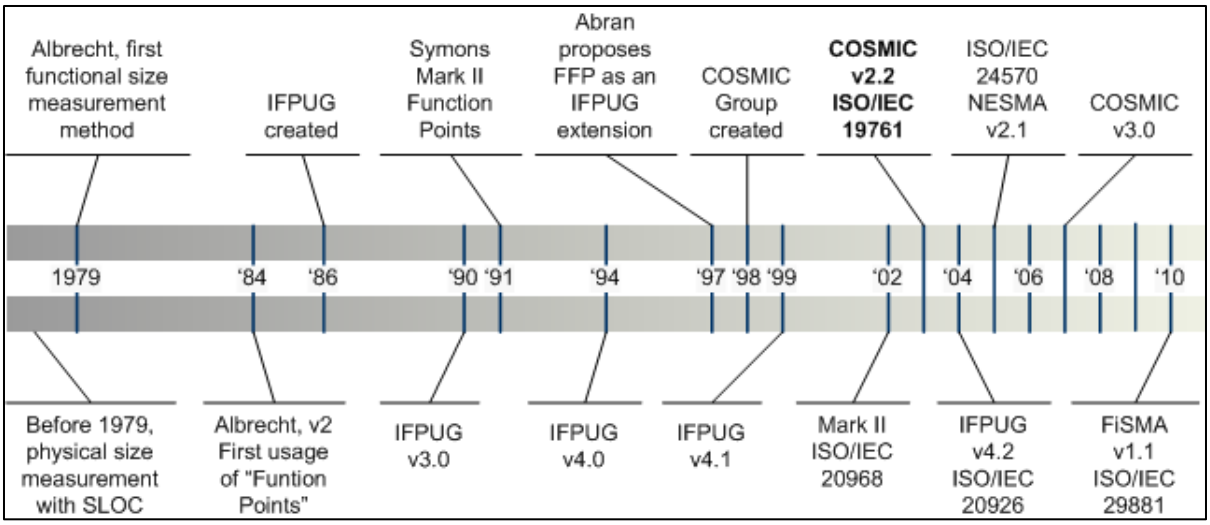

Figure 1.2 Software functional size methods publication history.

1.3.4 Functional size measurement: ISO standards

The International Organization for Standardization (ISO) has published several standards and technical reports related to generic concepts for functional size measurement methods, including:

1) ISO/IEC 14143-1:2007 Information technology -- Software measurement -- Functional size measurement -- Part 1: Definition of concepts (ISO, 2007): this document contains the mandatory definitions for software functional size measurement methods.
2) ISO/IEC 14143-2:2011 Information technology -- Software measurement -- Functional size measurement -- Part 2: Conformity evaluation of software size measurement methods to ISO/IEC 14143-1 (ISO, 2011a). This document contains criteria for evaluating a functional size measurement method to ensure compliance with the mandatory definitions for functional size measurement methods as prescribed in ISO 14143-1.
3) ISO/IEC TR 14143-3:2003 Information technology -- Software measurement -- Functional size measurement -- Part 3: Verification of functional size measurement methods (ISO, 2003a) describes the mechanism for verifying the compliance of a functional size measurement method with metrology verification criteria.

4) ISO/IEC TR 14143-4:2002 Information technology -- Software measurement -- Functional size measurement -- Part 4: Reference model (ISO, 2002a) containing examples of functional specifications of several software systems. This technical report presents sets of functional requirements that can serve as a reference to the different functional size measurement methods.
5) ISO/IEC TR 14143-5:2004 Information technology -- Software measurement -- Functional size measurement -- Part 5: Determination of functional domains for use with functional size measurement (ISO, 2004) contains software application domain definitions and criteria on which FSM methods apply or not.
6) ISO/IEC TR 14143-6:2006 Information technology -- Software measurement -- Functional size measurement -- Part 6: Guide for use of ISO/IEC 14143 series and related International Standards (ISO, 2006).
7) ISO/IEC 15939:2002 Software engineering -- Software measurement process (ISO, 2002b) provides a framework for implementing an organizational software measurement process to meet their information needs.
8) ISO/IEC 19761:2003 Software engineering – COSMIC: A functional size measurement method (ISO, 2011b).
9) ISO/IEC 20926:2009 Software and systems engineering – Software measurement -- IFPUG Functional size measurement method 2009 (ISO, 2009).
10) ISO/IEC 20968:2002 Software engineering -- Mark II Function Point Analysis -- Counting Practices Manual (ISO, 2002c).
11) ISO/IEC 24570:2005 Software engineering -- NESMA functional size measurement method version 2.1 -- Definitions and counting guidelines for the application of Function Point Analysis (ISO, 2005).
12) ISO/IEC 29881:2010 Information technology -- Systems and software engineering -- FiSMA 1.1 functional size measurement method (ISO, 2010).

Figure 1.3 illustrates these ISO standards and technical reports. The COSMIC method was chosen (in grey in Figure 1.3) for this research based on the researcher's motivation (see section 2.2). Also, as the chosen requirements document was describing functionalities of an

event-driven system (see section 3.4), the COSMIC method was the only method applicable in accordance with technical report ISO/IEC 14143-6:2006 (ISO, 2006).

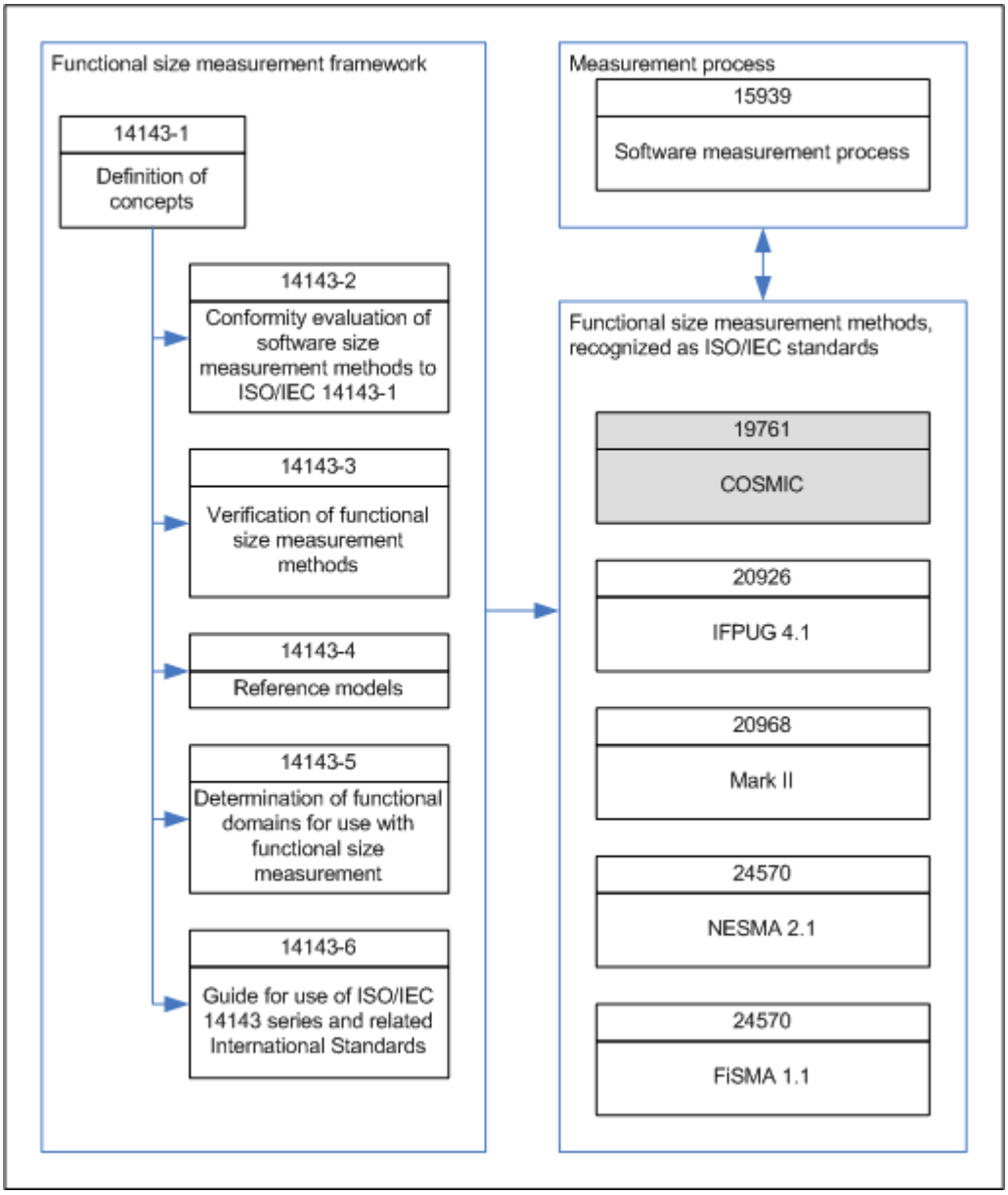

Figure 1.3 Functional size related ISO standards and technical reports.

1.4 The COSMIC method: ISO 19761

1.4.1 Overview of the COSMIC method

At a high conceptual level, software moves and manipulates data through a limited number of functional processes within its boundary. Functional processes can receive and send data to its functional users, which can be human users, other software applications, engineered devices, and a timer. When interacting with human users, the software is likely to go through input/output hardware, such as screen, keyboard, mouse, track pad, printer, etc. Functional processes can also read and write data to the persistent storage, regardless of the technology used, such as database systems, flat files, indexed files, eXtended Markup Language (XML) files, registers, etc. Figure 1.4 illustrates this generic flow of data from a functional perspective of any given software.

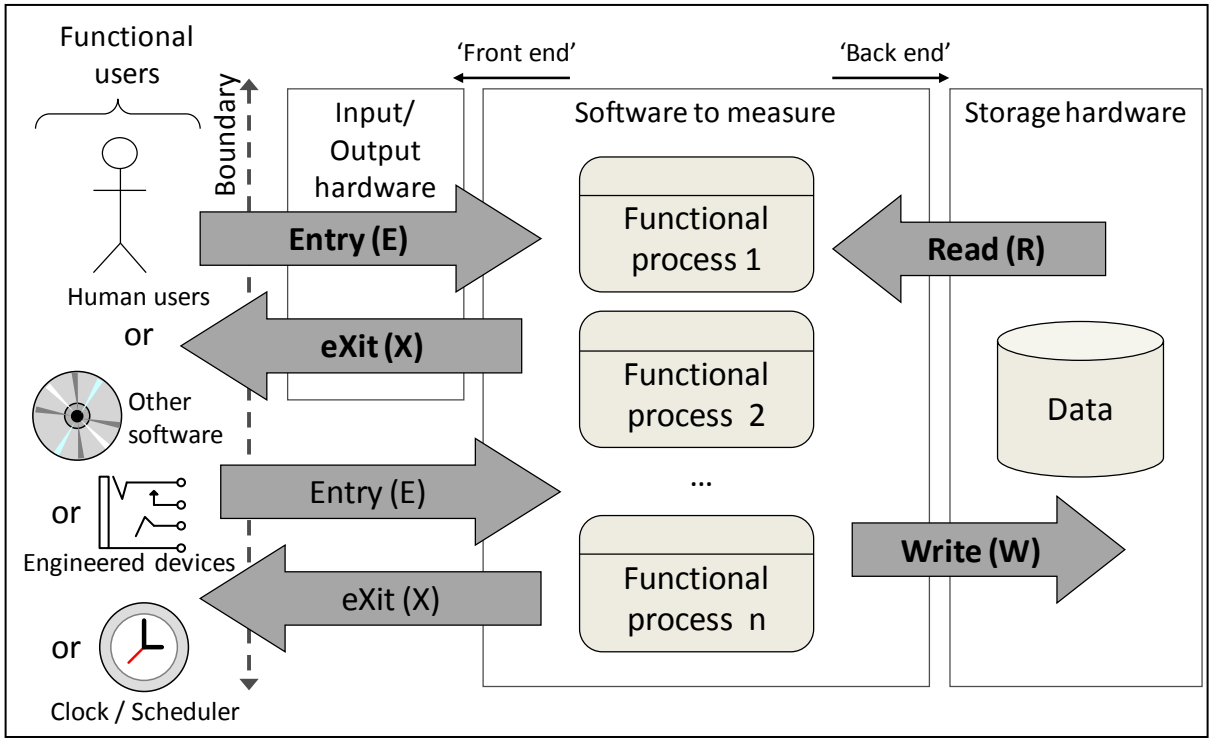

Figure 1.4 Generic software flow of data from a functional perspective
Adapted from (Abran *et al.*, 2003)

1.4.2 Data movements

The basis of measurement of the COSMIC 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 (DM) can be of four types: Entry (E), eXit (X), Read (R) or Write (W). 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.

1.4.3 Functional processes and their triggering events

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 a functional user. 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 DM: an Entry plus at least either an eXit or a Write.

1.4.4 The measurement unit of the COSMIC method

Each DM is counted as one COSMIC function point (CFP), which is the measurement unit of the COSMIC method.

1.4.5 Applicability and non-applicability of the COSMIC method

An excerpt from the COSMIC Method v3.0 Method Overview document (Lesterhuis and Symons, 2007) describes the applicability of the method as follows:

> "The COSMIC measurement method is designed to be applicable to the functionality of software from the following domains:
>
> - Business application software which is typically needed in support of business administration, such as banking, insurance, accounting, personnel, purchasing, distribution or manufacturing. Such software is often characterized as 'data rich', as it is dominated largely by the need to manage large amounts of data about events in the real world.
> - Real-time software, the task of which is to keep up with or control events happening in the real world. Examples would be software for telephone exchanges and message switching, software embedded in devices to control machines such as domestic appliances, lifts, car engines and aircraft, for process control and automatic data acquisition, and within the operating system of computers.
> - Hybrids of the above, as in real-time reservation systems for airlines or hotels for example."

There is no international consensus to apply the COSMIC method on mathematically-intensive software, such as expert systems, simulation software, self-learning software, weather forecasting systems, and other types of software characterized by complex mathematical algorithms or specialized and complex rules. Also, the COSMIC method has not been designed to measure software "which processes continuous variables such as audio sounds or video images, such as, for instance, in computer games, musical instruments, etc."

1.4.6 Measurers

In this research, a measurer is a person applying the COSMIC method.

1.5 Use of functional size measurement to improve requirements

1.5.1 Experience-based measurement results

An utilization of the application of the COSMIC method to improve the quality of the analysis model resulting from functional requirements has been documented (Nagano and Ajisaka, 2005). In their experiment, the proposed verification method is applied to functional requirements in isolation from peer reviews: they used a verification matrix which rows are user required functionalities, and columns are data groups. Up to five types of potential defects are detected from the quantitative measurement results:

1) Inconsistencies in the levels of abstraction from identified functional processes when the functional process size deviates significantly from the mean.
2) Weaknesses in the extraction of data groups when the functional size does not include read and write DMs.
3) Ambiguities related to data group identification when only Entry DMs are found.
4) Misidentifications of data groups that would be ‘effects from the system’ when only exit DMs are found.
5) Inappropriate naming of data groups when observing the same number of Entry and eXit DMs in more than one functional process, which may indicate that a data group has several roles.

The Nagano & Ajisaka proposed method has some challenges when the system is of significant size since the matrix becomes difficult to manage: a large number of functionalities combined with a large number of data groups provides a large matrix making it difficult to look for measurement defects. Therefore, it offers limited scalability. But, it was shown that it is already easy to apply simple verification mechanisms on small projects. These authors have deliberately rejected the application of peer reviews as being too dependent on reviewers’ skills. In addition, the Nagano & Ajisaka proposed method was

applied to embedded and real-time systems and authors claim that it may not be generalizable to other software domains.

1.5.2 Experience with use cases in addition to peer reviews

An empirical context was presented in (Baraby, 2006) where the functional size measurement with the COSMIC method was used in addition to peer reviews: the application of functional size measure raised defects different from those identified during peer reviews. However, the peer review approach used does not fall within any known formal method and the checklists used in (Baraby, 2006) contained items only directly related to use cases; furthermore, this review method was limited to use cases as a single and unique mechanism of requirements elicitation.

1.5.3 Relationships between quality of requirements and quality of FSM results

Two Japanese authors have looked into the relationship between requirements quality and the effort required to measure (Nishiyama and Furuyama, 1994). These authors have integrated FPA functional size measurement and the process of functional requirements elicitation and discussed the mutual influences between the requirements quality and the quality of measurement results. The authors assessed the quality of the requirements in their samples as either ‘good’ or ‘poor quality’, as if it was binary. However, these authors did not address the situation where requirements are not of good quality.

1.5.4 A cognitive approach for applying the COSMIC method

A cognitive approach to the application of the COSMIC method is presented in (Desharnais, 2003). This thesis addressed the problem of the consistency of measurement results of different measurers and presented cognitive aspects of the measurer. The author mentions a relationship between the quality of the requirements documentation and the quality of measurement results, but this issue is not addressed in details.

CHAPTER 2

RESEARCH ISSUES AND RESEARCH OBJECTIVES

2.1 List of research issues

This section presents a number of research issues identified from the analysis of the literature on requirements quality, review, measurement, and their relationship.

Issue # 1: Unknown quality of functional requirements

The quality of requirements is unknown until one or several reviews are applied.

Issue # 2: Uneven and unstable efficiency and effectiveness of functional requirements reviews

Nagano and Ajisaka (Nagano and Ajisaka, 2005) mentioned that review results are largely dependent on the reviewer skills. Weigers (Weigers, 2002, p. 49) identified several references with inconsistent reported efficiency and effectiveness results. This means that when a peer review approach is applied to requirements, requirements quality will get closer to the desired quality attributes even though there is still a significant risk that a number of defects were not identified or corrected, which defects would impact the rest of the software development cycle and maintenance.

Issue # 3: Effort to measure the functional size and measurement accuracy depend on the quality of the requirements

When a measurer comes across ambiguous or incomplete requirements, he should ask the authors of the requirement document for clarification in order to obtain FSM results of quality. Indeed, if the requirements are incomplete, the resulting functional size will be partial in relation to the software application that will be developed. And if requirements are ambiguous, the quality of FSM results will suffer since some functional processes, data groups, or DMs may be missed in the measurement process.

Issue # 4: Unknown effectiveness and efficiency of the COSMIC method to improve the quality of requirements

The literature review has not come up with references, with supporting data, on the effectiveness and efficiency of the COSMIC method to improve the quality of functional requirements.

2.2 Research motivation

During her professional practice as a software process improvement specialist, the researcher has had to measure since 2003 the functional size using the COSMIC method of many projects. Also, since 1996, she has had extensive experience in coaching and applying peer reviews. From her observations in industry, it appeared that applying the COSMIC method on functional requirements documents allowed identifying more defects in the same timeframe than if she would apply a peer review method on the same documents. But these observations were perceptions and qualitative: experimentation with other participants were needed to confirm or refute these perceptions. The researcher's motivation was therefore to understand and quantify to which extent the COSMIC method contributes to defect identification in functional requirements.

2.3 Research goal and chosen approach

The research goal is therefore to contribute to improving the quality of functional requirements by investigating the effectiveness and efficiency of applying the COSMIC method as a means for identifying defects and comparing the results with the application of a peer review technique.

2.4 Research objectives

To pursue this research goal, the research objectives selected are:

1) Determine experimentally the effectiveness (unit cost) of applying the COSMIC method in improving the quality of functional requirements compared to a peer review approach, when allotted limited effort as applied in the industry.
2) Determine experimentally the efficiency of applying the COSMIC method in improving the quality of functional requirements compared to a peer review approach.
3) Determine whether it is advantageous to include a measurer role in a peer review.
4) Determine the influence of defects on functional size.

2.5 Originality of the proposed research

This research work investigates the COSMIC functional size measurement method as a verification activity of functional requirements. Some researchers have mentioned it but none so far has carried out experimental or empirical research to document its effectiveness and efficiency.

2.6 Overview of research methodology

The research methodology selected to tackle the research goal and to reach the research objectives consists in a set of experiments. In the experimental research approach, it is necessary to define experimental protocols to ensure the quality of collected data and objectivity of the experimental results.

The experimental research methodology is structured into four phases:

1) Phase 1 will aim at refining the research methodology with a pilot project to test experimental concepts against criteria for quality of collected data and objectivity of results (see Section 3.1 for more details). Then, experimental material will be selected and prepared, in order to be ready for phase 2.
2) Phase 2 will focus on experiments where the same requirements document will be measured by measurers and inspected by inspectors. Experiments will be done with experts and with practitioners having limited experience in either applying the COSMIC method or inspecting documents.

3) In Phase 3, the experts' defects list will be used to apply corrections to the requirements document.
4) The updated requirements document will be used in Phase 4 with experiments focused on measuring the influence of defects on functional size. Two additional experiments will be conducted, one with expert measurers, the other with measurers having limited experience.

Figure 2.1 presents an overview of this research methodology, including activities, artefacts (inputs and/or outputs), and research outcomes. Numbers in small fonts represent section numbers of this thesis or appendices (App#) hereof.

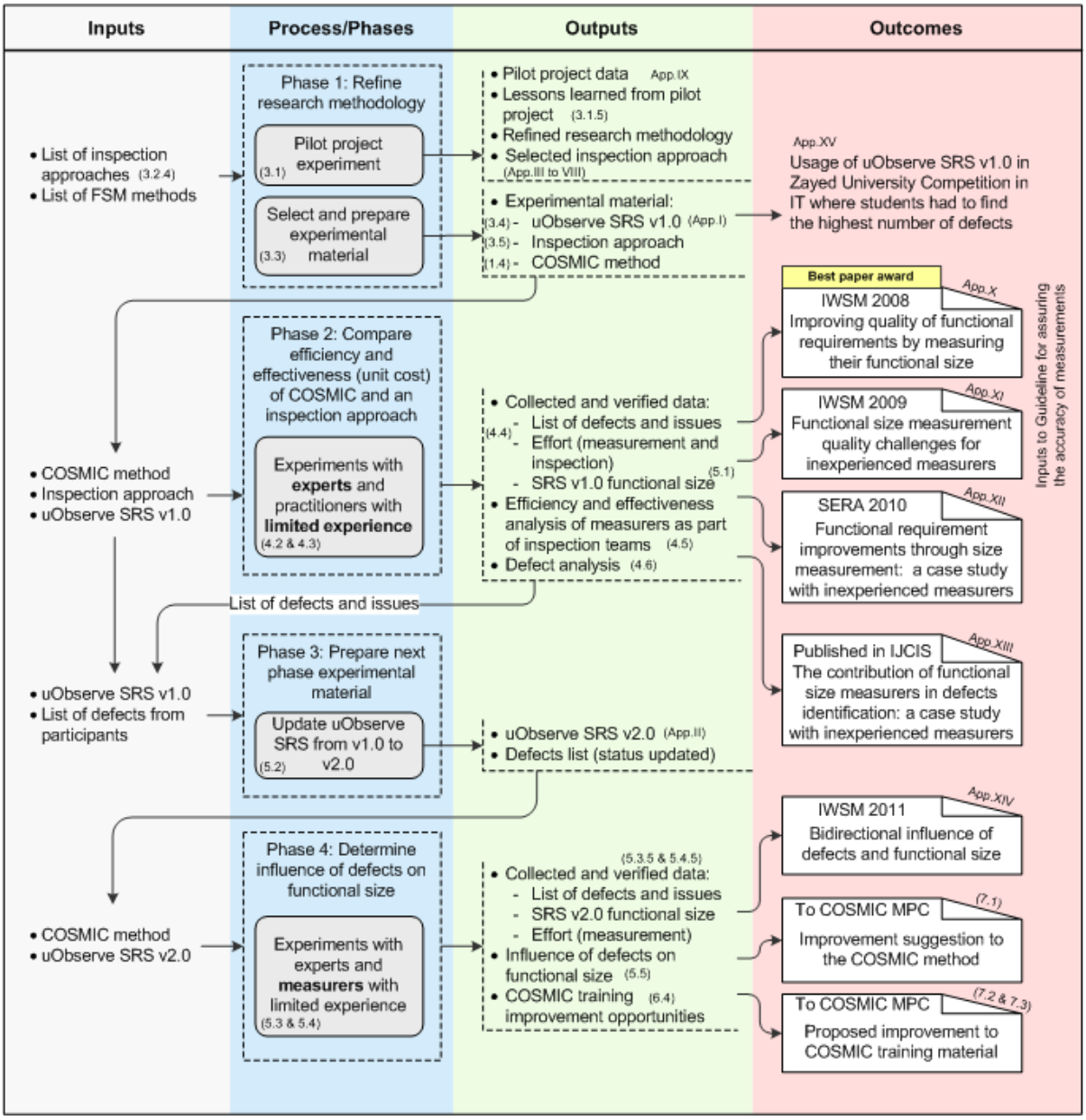

Figure 2.1 Overview of the research methodology

CHAPTER 3

PHASE 1: REFINE RESEARCH METHODOLOGY

3.1 Pilot project experiment

3.1.1 Purpose and objective of the experiment

Phase 1 consist of a pilot project with software engineering undergraduate students in order to refine the research methodology. The objective of the pilot project was to analyze any flaws and risks from the applied pilot project methodology to refine this research methodology for the subsequent research phases.

3.1.2 The requirements documents

The requirements document used for this research project came from the following source: in an undergraduate software requirements course, the professor had selected a ConOps document (referred to as the 'Vision' document in that course) describing portions of an autonomous submarine, known as S.O.N.I.A. (Système d'opération nautique intelligent et autonome) that performs a series of underwater tasks to accomplish missions without human intervention. This S.O.N.I.A. submarine had digital cameras, sound probes, hydrophones, and several motors allowing autonomous navigation. From that chosen ConOps document, each team of students had to write the SRS for the submarine vision editor software. Ten teams had written ten different SRS documents describing functional and non functional requirements that would be compliant to elicited needs in the ConOps. Eight teams had written use cases to describe functional behaviour of the vision editor.

For the pilot project SRS documents from two teams were discarded because they contained scarce or no functional requirements.

3.1.3 The participants

3.1.3.1 The inspectors

The inspector-participants for the pilot project experiment were twenty undergraduate students who were attending the software requirements course. They were grouped into eight teams of two, one team of one, and one team of three students. They all had limited experienced in applying inspections or peer reviews.

3.1.3.2 The measurer

The researcher, skilled and experienced with the COSMIC method, was the sole measurer for the pilot project.

3.1.4 The experiment steps

Training session and training material on the CRIM inspection approach were given by the researcher to all inspector-participants. Then each team provided the SRS document they had written to another team and an identical SRS copy to the researcher. Each team applied the inspection approach on the SRS document they received from another student team, collecting defects, issues, and effort data on the inspection form.

Meanwhile, the researcher applied the COSMIC method to each SRS document, identifying defects and issues, and measuring effort spent performing their measurement and defect identification tasks.

Once done, a quick logging meeting (between 15 and 20 minutes) was held with every team to explain critical and minor defects, ensuring with team members that these were real defects and of the adequate type.

Finally, the researcher collected and analyzed defect, size, and effort data from the participants (see Appendix IX) and presented to them a result summary.

3.1.5 Lessons learned from the pilot project and methodology requirements

At the end of this pilot project, lessons learned were derived from several identified weaknesses, allowing the researcher to define a research methodology that would provide greater objectivity and quality of the results:

1) The pilot project methodology had a risk of biased data due to the direct involvement of the researcher in providing the measurement data. Ideally, measurement and inspection results must be independent from the researcher to avoid any biased data to be analyzed. Furthermore, to ensure greater quality of measurement data, experts in the COSMIC method must be involved.
2) Experiment data was handwritten on the inspection form and on the SRS document. This data required to be collected in a spreadsheet for analysis. Mistakes can be made during data entry, which would corrupt data and related analysis. Therefore, any collected data that is entered in a database or spreadsheet should be verified by a different person than the researcher to ensure data integrity with data sources.
3) Eight different SRS documents were inspected and measured, all describing a software solution to the same problem domain and needs. There were only one to three inspectors per SRS document (average: two inspectors per SRS), and only one measurer per SRS document. To obtain a substantial quantity of data to analyze, a significantly large sample of SRS documents would be required, and on which the same methodology must be applied. Obtaining a large sample of SRS documents was unrealistic. Attempts to obtain SRS documents from the industry failed, mostly because the industry managers contacted were not willing to share their requirements, even if confidentiality agreements were proposed. Other attempts in finding SRS documents from academics posed similar problems: lack of SRS documents or unwillingness and shyness of their authors to make it available in a research context, once they understood that more than ten people would inspect these documents identifying a potentially large number of defects. For practical

considerations, the number of SRS documents must be kept as small as possible, ideally only one, to allow gathering significant data for analysis purposes. Selecting only one SRS may affect the applicability of the proposed approach to different applications types or different means of requirement documentation.

4) With an average of two inspectors per SRS document and only one measurer, there was a risk of too many variables and a lack of generalization power. The research strategy adopted was then to limit the experiment scope to a single SRS document, but to have a number of experiment participants as large as possible to provide meaningful data analysis.
5) The Gilb & Graham inspection approach with CRIM's adaptations was consistently applied by all participants after they had participated in a relatively short training workshop (between one and two hours). Therefore, a training workshop along with inspection training material should be given to all inspection participants to ensure repeatability of the inspection method.
6) The inspection method was applied in a 'serial' mode (see Section 3.5.1 for more details) for the pilot project so that participants, who all had limited experience in performing peer reviews, would learn from their peers about identifying defects. But they had a week to perform their individual inspection, which was not likely to happen in controlled environment experiments. Therefore, due to time constraints for single experiments, the 'parallel' inspection mode was selected for the subsequent research phases.

3.2 Overview of the experimental research framework

From the lessons learned from the pilot project, and to help structure the experimentation process, the Basili (Basili, Selby and Hutchens, 1986) research framework has been used. This research framework includes four phases:

I. Experiment definition, where the problem to solve and the research scope are laid out.
II. Experiment planning, where experimental design, criteria, and measurement are selected.
III. Experiment operation, where preparation, execution, and data analysis are applied.

IV. Experiment interpretation, where interpretation is explored along with potential extrapolation of results and industry impacts.

Figure 3.1 shows a summary of this research experimental framework.

I. Definition					
Motivation	Object	Purpose	Perspective	Domain	Scope
Verify the contribution of the COSMIC method to identify defects in functional requirements	Functional requirements	Evaluate quantitatively	Requirements Inspectors COSMIC Measurers	Requirements engineering (functional requirements) and measurement (sizing)	One requirements specification document
II. Planning					
Design		Criteria		Measurement	
Pilot project Statistical analysis methods		Defect types and categories Participant types and experience levels		Number of defects per participant Number of unique defects per team Effort Functional size	
III. Operations					
Preparation		Execution		Data analysis	
Select and print material Call for participation Train participants		Collect data Verify data		Preliminary analysis: experts and practitioners with limited experience separately Formal analysis: • combined data from Phase 2 experiments • data from Phase 4 experiments	
IV. Interpretation					
Interpretation context		Extrapolation		Impact	
Within one inspection approach, one set of SRS document, multiple inspectors and multiple measurers and one functional sizing method		Identification of defects while measuring		Application of the COSMIC method as a means to identify defects	

Figure 3.1 Research experimental framework
Based on (Basili, Selby and Hutchens, 1986)

3.3 Selection and preparation of experiment material

This section presents the material selected for the experiments which consists of three components:

1) A requirements document (The uObserve SRS document, section 3.4).
2) An inspection approach, including training material and related artefacts (section 3.5).
3) The COSMIC method (section 1.4).

The COSMIC method was chosen because its usage in industry led the researcher to initiate this research project. The COSMIC method did not require preparation for our experiments. Instead, selection of participants was applied in order to classify e their experience into ‘expert’ or ‘limited’.

3.4 The uObserve SRS

Based on the lessons learned from the pilot project (see section 3.1.5), the objective was to select an SRS document which structure and content were likely to be of relatively good quality, but that may still contain several defects with regards to the desired quality attributes (see section 1.1.4). As an input to the experiments, it was required that the chosen software requirements specification (SRS) document had the following characteristics:

- Compliant with IEEE-Std-830 for its structure and content in order for experiment participants to be somewhat familiar with the SRS structure.
- Compliant with UML 2.0 (Arlow and Neustadt, 2005) for the use case diagram, the behavioural state machine and use case details, as this description format is largely used in the industry and would probably be familiar to experiment participants.
- Contain text not exceeding 3000 words (the equivalent of 10 pages of 300 words each) in order to be inspected within an hour at a rate of 10 pages per hour.
- The software systems described in the SRS had been successfully developed and tested, in order to avoid a purely theoretical SRS as it would describe functioning software.

- The majority of the described functional processes are meant to be used by human users, not electronic or engineered devices, in order for any experiment participant to understand how a user would interact with the system.

Effort has been made at early phases of this research project to find an SRS document from the industry. This approach was unsuccessful mainly because the industry requires confidentiality over their software product requirements and the chosen SRS document was to be used in experiments involving at least 20 participants. Also, industry SRS documents tend to be of larger size than what was aimed for. Therefore, the researcher looked at small software systems developed as part of graduate study projects for any of ÉTS's research laboratory.

The uObserve system, developed in 2004 for the *Laboratoire d'environnements de synthèse et interfaces avancées* (LESIA), complied with all required characteristics. However, the SRS document was written in French and needed translation in English for experimental usage.

3.4.1 Preparing the uObserve SRS for experimental usage

The first step in preparing the uObserve SRS document was to obtain the permission from its authors to use it in this research project. They generously accepted. The French version of uObserve SRS had been peer reviewed by the team members prior to the software development activities.

The second step was to translate the uObserve SRS into English, including textual, graphical, and format parts. Then to ensure that the translation had not introduced too many defects, the resulting English SRS was peer reviewed by an industry colleague. Other minor issues were also identified and corrected. Then, it was baselined to version 1.0 (v1.0, see Appendix I) and ready to use in the research experiment.

3.4.2 The uObserve system overview

The uObserve SRS describes an event-driven software tool supporting usability testing of user interfaces. It is composed of two software applications in a client-server architecture. The client software is a Java library called uSpy that must be included in any Java software that has a user interface for which usability testing is required. This means that the uSpy software is a component that must be included and linked with the other Java software when compiling it. The uSpy component main role is to spy upon the human user's actions by capturing all mouse and keyboard events, which are automatically generated by the client workstation operating system, and to send those events to the server software, named uSleuth, over the Local Area Network (LAN).

The uSleuth software, running on the server workstation, records and plays back an audio/video file using a camera installed (plugged) directly on this server. The camera was installed on the server as a design constraint since it required processing resources that would have affected the client's workstation software performance, where the context of usability testing required the least amount of disruption on the client's workstation. The uSleuth software receives mouse and keyboard events from the uSpy software component, records these events as a log of events, and synchronizes these events with the audio/video frames. Figure 3.2 illustrates a typical uObserve operational system setup.

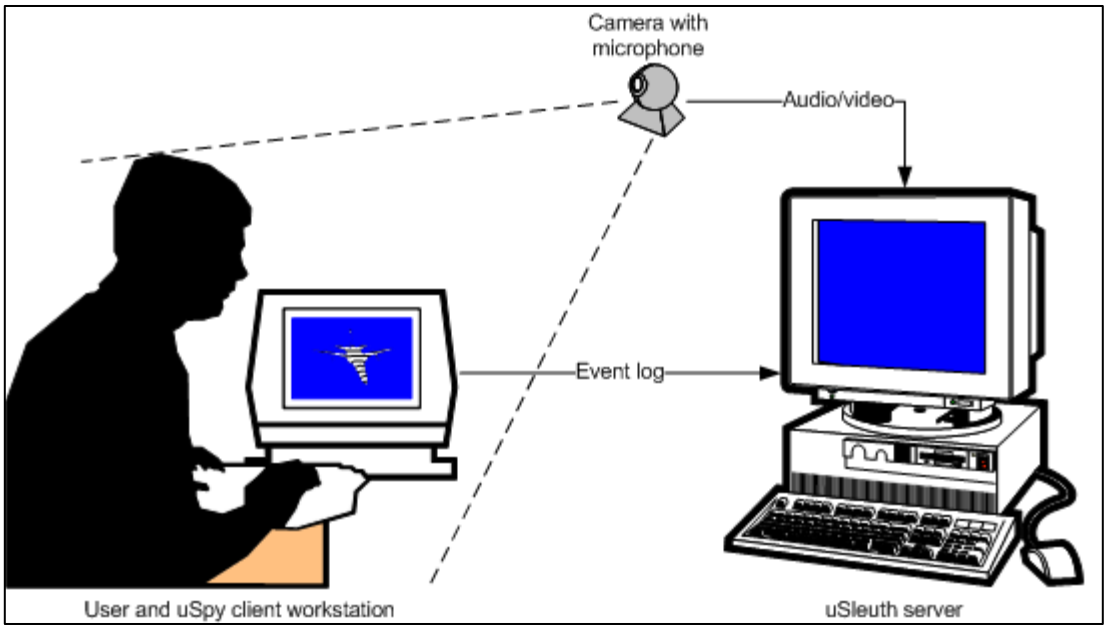

Figure 3.2 Typical uObserve operational system setup

The uObserve system is a process control system which task is to keep up with events happening in the real world. It is part real-time when recording and receiving events and part data centric when playing back previously recorded experimental data.

3.4.3 The uObserve SRS structure

The SRS document was labelled 'uObserve Software Specification' 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 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).

3.5 The inspection approach

The CRIM inspection approach was used in the experiments. This CRIM inspection approach consists of several adaptations from the Gilb & Graham approach. This CRIM inspection approach has been applied successfully in a Canadian defence organization (DND, 1997-2000) more than 2000 times over a four years period and numerous times in other Canadian organizations since 2001 (Trudel, 2002; 2003; 2007). It consists of seven steps, along with exit criteria, as shown in Figure 3.3.

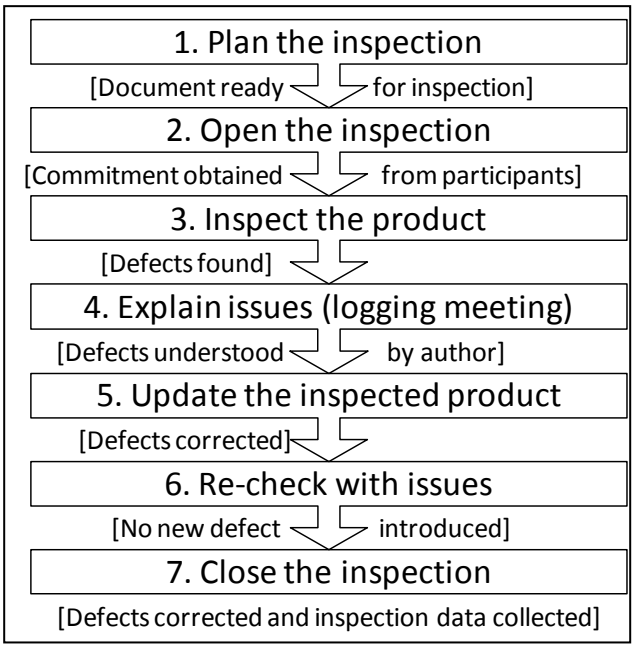

Figure 3.3 Steps of the CRIM inspection method

While these seven steps are generally applied in the software industry, only the first four were relevant in experiments using uObserve SRS v1.0 document. Step 5 was done prior to experiments using uObserve SRS v2.0, which stood as the sixth step. Then, the researcher and colleagues verified collected inspection data prior to data analysis, as the seventh step.

3.5.1 Inspection modes

Two inspection modes are defined in the CRIM inspection method: ‘parallel’ or ‘serial’. In ‘parallel’ mode, every inspector has his own copy of the document to inspect and they inspect the product at the same time. In ‘serial’ mode, only one copy of the product 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 experiment time constraints, the ‘parallel’ inspection mode was applied.

3.5.2 Inspectors

An inspector is a person participating in an instance of the inspection method, specifically performing step 3 and participating in steps 2 and 4.

3.5.3 Inspection material

This inspection method required material to be consistently applied in the experiments. Copies of the following artefacts were shown or distributed to experiment participants as part of their inspection workshop training package:

1) Presentation material for the inspection workshop: a set of slides used by the presenter to teach this specific inspection method to experiment participants (see Appendix III).
2) The inspection method summary: a graphical overview representation on an 8.5” x 11” sheet (see Appendix IV).
3) The detailed inspection method: a graphical detailed representation of the steps, objectives, defect types, inputs, outputs, activities by role, and exit conditions, all on a single 11” x 17” sheet (see Appendix V).
4) A list of defect and issues types (see Table 3.1). Improvement suggestions and questions are considered as issues, not as defects. However, a question may be transformed into a critical or minor defect during the logging meeting, depending upon the nature of the question and its related answer.

5) A list of definitions for defect categories (see Table 3.2). Defect categories were defined for analysis purposes, since measurement should primarily be dealing with the functional description of the system to develop.
6) An inspection form: a paper form designed to collect data on the current inspection, such as effort spent per step and total effort, and the number of defects and issues per type (see Appendix VI).
7) A list of peer review rules: three sets of rules were defined (generic, requirements, and design). The generic and requirements rules (Table 3.3) (Gilb and Graham, 1993) were relevant and used to objectively identify defects (see Appendix VII).
8) A list of peer review roles: applying different roles in a single inspection provides different viewpoints among inspectors who are likely to find more defects due to different perspectives on the product under review (see Appendix VIII).

Table 3.1 Definitions for defect and issue types

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

Table 3.2 Definitions for defect categories

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 3.3 List of inspection rules

Generic rules (applicable to any type of information)
1) COMPLETE: Information shall be complete for its intended purpose.
2) CLEAR: Information shall be clear for its intended reader.
3) CORRECT: Information shall be free of errors.
4) CONSISTENT: Information shall be consistent within its content, and with other information it references or from which it derivates.
5) RELEVANT: All information shall be pertinent to the section or context it is used.
6) BRIEF: Information shall be presented in a succinct manner.
7) REFERENCE: All references contained in the information shall be correct and verifiable.
8) RISK: Any known risk that can be derived from a statement shall clearly be cited.
Requirements Rules (applicable to written requirements)
1) TESTABLE: A requirement shall be testable by any means.
2) ELEMENTARY: Requirements shall be stated at the lowest possible level. A requirement shall contain only one testable item. As a guideline, if two or more things have to be tested in order to validate the requirement, then it is not expressed at the elementary level[1].
3) NEED: A requirement shall be stated in term of final needs, not perceived means (state the « What », not the « How »).

[1] In a use case description, an ‘elementary level” element would refer to a single step in a scenario.

CHAPTER 4

PHASE 2: COMPARE COSMIC AND INSPECTIONS

4.1 Overview of Phase 2 experiments

In Phase 2, three experimental sessions were held in order to gather data related to the first three research objectives (Figure 4.1). A total of 35 practitioners participated in these experiments: 17 inspectors and 18 measurers.

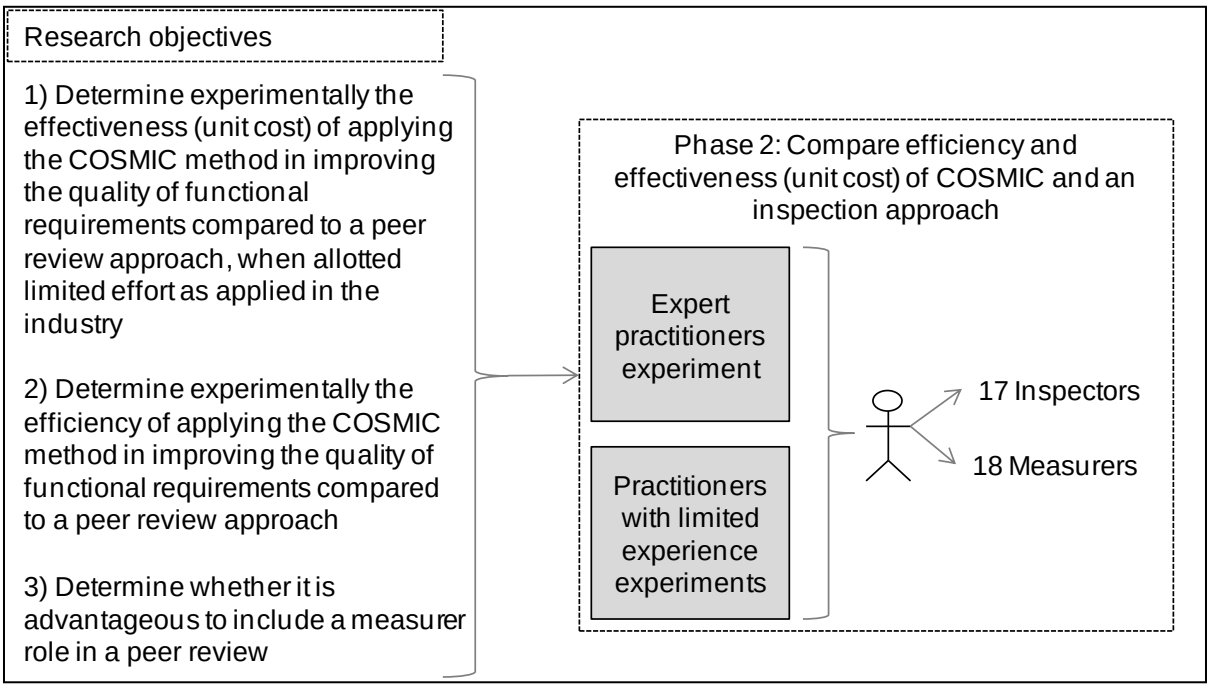

Figure 4.1 Overview of Phase 2 experiments.

The first experimental session, held in November 2007, targeted expert practitioners in order to establish a baseline of data from experts. This first experiment was a workshop included within the program of the 2007 edition of the International Workshop on Software Measurement (IWSM), a conference for specialists and experts in the field of software measurement.

The second experimental session was held in May 2008 and targeted practitioners with limited experience who were graduate students in a software measurement course at École de Technologie Supérieure (ETS). One expert inspector participated in this experimental session and her data was added to experts' data from the first experiment.

To gather more data, a third experimental session was held in August 2010 as part of a software measurement symposium for software engineering graduate students interested in software measurement. One expert measurer participated in this third experimental session: experimental data he provided was added to the experimental data from the first experimental session involving experts.

4.2 Description of the experimental protocols

4.2.1 Objectives of the experiments

The experimental sessions of Phase 2 aimed at the first three research objectives:

1) Determine experimentally the effectiveness (unit cost) of applying the COSMIC method in improving the quality of functional requirements compared to a peer review approach, when allotted limited effort as applied in the industry.
2) Determine experimentally the efficiency of applying the COSMIC method in improving the quality of functional requirements compared to a peer review approach.
3) Determine whether it is advantageous to include a measurer role in a peer review.

4.2.2 Purpose of experiments

The general purpose of these experiments was to gather data from experts and also from participants with limited experience in order to compare results and observe similarities and differences based on practitioners' claimed experience.

The purpose of the first experiment was to involve 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 document. Special care was taken to get experienced practitioners in FSM and experienced inspectors and requirements writers in participating to this experiment.

The purpose of the second experiment was to involve people with limited experience in peer reviews as well as in using the COSMIC method. The purpose of the third experiment was to gather supplemental data to better support this research project.

4.2.3 The requirements document

The uObserve SRS v1.0 was used in all three experimental sessions. One copy for each participant was printed.

4.2.4 The experiment steps

At a high level, the same steps were applied in all three experimental sessions. Some variations occurred in the details of these steps, and are discussed next. The experimental sessions consisted in the following steps:

1) Prepare experiment
    a) Prepare material for the experiment

        The experiment material included the uObserve SRS v1.0 and all artefacts from the inspection approach. There was no specific preparation to make about the usage of the COSMIC method as participants who volunteered to apply the method had declared knowing it. Thus, participants from the first experiment were known COSMIC experts.

b) Recruit participants

A call for participation to the first experimental session was included within the Call for participation to the MENSURA-IWSM-2007 conference, knowing that a mix of industry and academic experts attend this event each year. Participants from the second experimental sessions were volunteers from a graduate software measurement course where their professor generously arranged the schedule to make room for this experiment. They were not graded for their participation and were free to leave the classroom if they did not want to participate. Participants from the third experimental session registered into a software engineering symposium at ETS in August 2010. The experimental workshop was part of the symposium program and they all volunteered to participate. These participants were all graduated from software engineering programs, with various levels of industry experience. A subset of participants were students of master or doctorate software engineering programs and were not graded for their participation, which was volunteer work. Everyone was free to leave the room before experiment begins.

c) Qualify participants

At the very beginning of every experimental session, the researcher explained the objective and purpose of the research and the specific experiment as well as its major steps.

In order to determine which participant would apply the COSMIC method and which ones would apply the inspection approach, participants were asked if they knew the COSMIC method well enough to apply it. Those who did were assigned as measurers. The others were assigned as inspectors. All inspectors were asked to provide their experience level in applying peer reviews in months or years. All measurers were asked to provide their experience level in applying the COSMIC method (in months or in number of software they had measured) and if they were certified COSMIC entry-level. More details on participants are provided in section 4.3.

2) Provide training on the inspection approach

A one-hour training session was provided to all participants on the inspection approach, the rules, the roles, and the behaviours to expect and to avoid from inspection participants (inspection leader, author, and inspectors). Emphasis was put on the rules and defect types to ensure participants understood what would be considered as defects (i.e. when one of the rules is violated) or issues and their related type classification.

They all had access to the list of rules, the list of roles, and the definitions of defect and issue types.

3) Apply the inspection approach

a) Plan the inspection

Table 4.1) were a subset from the provided list of roles. Assigning several inspector roles aims to maximize defect identification as different perspectives are being applied.

Table 4.1 Required inspector roles and their definition

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).

The inspection scope was defined as sections 2 and 3 of the SRS document with a priority on section 3.2 containing the functional requirements, which text size was measured at 1735 words (corresponding to an equivalent of 5.8 pages of 300 words).

The whole document contained 3563 words, including titles. Thus, planned individual checking effort was set to 1 hour and 10 minutes (70 minutes) based on an inspection rate of 5 pages per hour. The source documents for section 3.2 were the SRS itself (sections 1, 2, 3.1, and 3.3) and applicable standards (IEEE-Std-830 and UML 2.0).

For the first two experimental sessions, the one-page inspection form was printed with information about the SRS document to inspect and the planned effort. However, only one line of defect summary data was captured (participant name, role, number of defects per type, and effort). For the third experimental session, as only one line was used on the form by participants, the information about the document and the planned effort was written on a white board and they were asked to provide their defect summary data on the first page of the SRS document. This allowed for data to be kept with every SRS copy.

b) Open the inspection

A brief SRS overview was provided to inspectors and measurers. Instructions were given to set a type (C, M, S, I or Q) and a category (F, N, or U) for every defect or issue. Inspectors were asked to select a role within the list of roles and to write this role on their inspection form. Then, they were asked and agreed to play their selected role.

c) Inspect the product

At this point in experiment sessions, the clock started clicking for individual effort. The inspectors began their checking task and the measurers began their FSM task.

Inspectors did their individual checking, playing their selected roles the best they could and using the list of rules as their basic checklist. Defects and issues were identified and noted on the copy of the SRS document of each inspector, along with their respective type and category. Inspectors stopped the checking activity when they were convinced they had completed the required verification. Then, each inspector

compiled the number of defects per type and reported this data on the inspection form or on the front page of their SRS copy.

In the first and second experimental sessions, participants measured their individual checking effort and compiled it on the inspection form. In the third experimental session, the researcher compiled the actual effort as all participants completed their task in 60 minutes +/- 2 minutes. This was done to alleviate the task of participants who could concentrate on defects, knowing from the first and second experimental sessions that there would be limited effort difference among participants.

d) Explain issues (logging meeting)

In the first experimental session, a one-hour logging meeting was conducted with the independent inspection leader and the three inspectors to describe every identified defect and issue. The objectives of the logging meeting were:

i) To confirm whether identified defects and issues were indeed defects and issues in accordance with the provided type definitions;

ii) To assign or change the defect or issue type if required (e.g. from Question to Minor or Critical defect);

iii) For the researcher, to understand all confirmed defects and issues with the purpose to be able, at the next edit step, to apply appropriate corrections and, if required, a type reclassification.

Defects of the spelling/syntax type were voluntarily skipped since explanations were not required. At the end of the logging meeting with the inspectors, the measurers described the portion of their identified defects that they considered the most important. Finally, all SRS copies with their hand-written defects and issues were given back to the researcher.

In the second and third experimental sessions, the number of participants was unsuitable for a group logging meeting. Therefore, participants were asked to give their SRS copy to the researcher and logging meetings were later conducted with the assigned inspection leader of each experimental session. In the second experimental session, the inspection leader was the most experienced inspector from that experimental session. In the third experimental session, the inspection leader was an experienced colleague who participated in this third experimental session. These differences in the data collection approach are believed to have no impact on the collected defect data, but they had a positive impact on the effort required due to the number of participants. The effort per experiment for explaining issues and defects was later divided by the number of items discussed to obtain an average number of minutes per defect and issue for that step.

4) Measure the COSMIC functional size

The inspection training provided guidance on defect types and categories to measurers, who attended the session as well. When the researcher handed a printed copy of the SRS document to each measurer, measurers were asked to apply the COSMIC method and to identify any defect and issue, along with its respective type and category. While inspectors were inspecting, 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 DMs. DMs were added to provide the functional size of every functional process. These measurement details were either written directly on the SRS copy or on separate blank sheets of paper.

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 form or on the first page of their SRS copy: effort time to measure and to identify defects, number of defects per type, and software functional size.

5) Compile experimental data

a) Defects and issues log

Defects and issues were logged by the researcher in a spreadsheet with the following parameters:

- Location (page #, section #, paragraph #, and line #).
- Description of the defect or issue.
- Defect or issue type (C, M, S, I, or Q).
- Category (F, N, or U).
- Inspectors' identification code.
- Number of inspectors (when more than one identified the same defect or issue).
- Measurers' identification code.
- Number of measurers (when more than one identified the same defect or issue).
- Status (Open, Resolved, or Rejected).
- Comments from the researcher.

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.

Also, several measurers had written down notes explaining their measurement results that were based on some measurement assumptions. Therefore, the hand-recorded notes were recorded in the same spreadsheet with the type 'A' for measurement assumption as described here:

> A measurement assumption is not a defect but could be derived from a defect and takes the form of written notes from measurers describing how they applied the measurement activity on a specific requirement within the SRS.

These measurement assumptions were not defects, but they were describing how the measurement result was obtained, derived from already identified defects. These assumptions could be true so would be their related measurement result, or they could be wrong and so would be their related measurement result.

b) FSM detailed data

The following FSM detailed data was captured in a spreadsheet:

- The functional processes.
- Their relevant data groups.
- For each measurer:
  - i) DMs per data group.
  - ii) The size per data group.
  - iii) The size per functional process.
  - iv) The number of functional processes identified.
  - v) The 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 approach was entered separately.

6) Review experimental data with participants or inspection leader

a) Data quality assurance of the first experiment

Individual data of the first experiment were extracted from the spreadsheet per participant and sent to each of them 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 extracted from the spreadsheet against the scanned copy of their hand-written commented SRS. Data were hidden from one another to avoid any bias or influence. This step was made to ensure that data analysis

would be made with unbiased and verified data. All participants sent review feedback with either minor changes or no comment.

b) Data quality assurance of the second experimental session

Data of the second experimental session were reviewed with the most experienced inspector who acted as the inspection leader. Detailed comparison of defects and issues captured in the spreadsheet were reviewed against hand-written SRS copies of every participant to ensure correctness of data. The researcher and the inspection leader made sure they had common understanding of each defect and issue, and that each defect and issue was of the appropriate type. The researcher and the inspection leader also verified that measurement data were adequately captured.

c) Data quality assurance of the third experimental session

Data from the third experimental session were independently reviewed by a participant who acted as the inspection leader. This review was to ensure accuracy of captured data with physical evidences (SRS copies with hand-written defects and issues, measurement data, and observed effort).

7) Analyze experiment data

a) Intermediate data analysis

After the first experimental session, data was analyzed: preliminary results indicated that measurers had a significant value-added in finding defects while applying the COSMIC method (appendix X).

From the second experimental session, data was also analyzed and two types of findings were observed:

- Participants with limited experience in applying the COSMIC method tend to make similar measurement mistakes (appendix XI).
- Measurers with limited experience add value in defect identification over inspectors of limited experience (appendix XII).

b) Consolidating data

With a total of 35 participants, there was a need to consolidate data from all three experimental sessions. In fact, the purpose of the third experimental session was, up to a point, to gather more data for data analysis and it was planned that this data would be merged with data from the first and second experimental sessions.

Consolidating defect and issue data had an impact on defect types. As new duplicates were identified and merged into unique defects, the original defect type may have been different and, as a guiding principle, the merged defect type was the one with the highest impact (e.g. Critical over Minor, or Question transformed into Critical or Minor defect).

c) Analyzing consolidated data

In industry, FSM is more likely to be made by a single measurer. Therefore, experimenting with eighteen measurers represented eighteen different experimental data sets.

From the inspection point of view, the industry generally applies from two to four inspectors for a single inspection of a requirements document. Therefore, assumptions had to be made whether there would be two, three or four inspectors in an inspection team and then replace or add one measurer per inspection team.

4.3 Experiment participants

4.3.1 The inspectors

A total of 17 inspectors participated in the three experimental sessions. A unique identifier has been assigned to each inspector. From the experience description each inspector provided, an expertise level was assigned based on the following criteria:

- Limited: less than 48 months of review experience.
- Expert: 48 months and more of review experience.

Table 4.2 provides a list of these inspectors with their experience description.

Table 4.2 List of inspectors

Inspector ID	Experimental session	Expertise level assigned	Experience description	# Months of experience assigned
A	May 2008	Limited	Limited	3
B	May 2008	Limited	Undergraduate courses only	3
C	May 2008	Limited	Limited, few reviews only	3
D	May 2008	Limited	(Unspecified, Master degree student. Assumed 6 months)	6
E	May 2008	Limited	3 years as document reviewer	36
F	May 2008	Limited	10 years as SW developer, 6 months as reviewer	6
G	May 2008	Limited	8 years as SW developer, 2 years reviewer	24
H	May 2008	Limited	Less than a year	6
I	May 2008	Limited	4 years in SW, half in reviews	24
J	Aug. 2010	Limited	6 months reviewer	6
K	Aug. 2010	Limited	4 months reviewer	4
L	Aug. 2010	Limited	13 years in SW, 1 year reviewer	12
M	Aug. 2010	Limited	Over 13 years in SW, 1 year as reviewer	12
N	Nov. 2007	Expert	15 years in SW, 4 years as reviewer	48
O	Nov. 2007	Expert	19 years in SW, 13 years as reviewer	156
P	Nov. 2007	Limited	8 years in SW, 1 year reviewer	12
Q	May 2008	Expert	29 years in SW, 4 years reviewer	48

Based on assigned expertise level, there were three expert inspectors and 14 inspectors with limited experience.

4.3.2 The measurers

A total of 18 measurers participated in the three experimental sessions. A unique identifier has been assigned to each measurer. From the experience description provided by each measurer, an expertise level was assigned based on the following criteria:

- Limited: 3 years or less of experience using the COSMIC method or less than 5 software applications measured, and could be certified COSMIC entry-level or not.
- Expert: 3 years or more of experience using the COSMIC method, must be certified COSMIC entry-level, and participate as an active member of the COSMIC Measurement Practice Committee (MPC) or has been involved as a co-author of the COSMIC method.

Table 4.3 provides a list of these measurers with their experience description.

Table 4.3 List of measurers

Measurer ID	Experimental session	Expertise level assigned	Experience description	COSMIC Entry-level Certified?
C	Nov. 2007	Expert	Co-author of the COSMIC method, member of the MPC	Yes
F	Nov. 2007	Expert	Experienced practitioner, member of the COSMIC MPC	Yes
H	Nov. 2007	Expert	Experienced practitioner, member of the COSMIC MPC	Yes
L	Nov. 2007	Expert	Experienced practitioner, member of the COSMIC MPC	Yes
J	Aug. 2010	Expert	Co-author of the COSMIC method, member of the MPC	Yes
M	Aug. 2010	Limited	1-5	No
N	Aug. 2010	Limited	3 years with COSMIC	Yes
O	Aug. 2010	Limited	2 years with COSMIC	Yes
P	Aug. 2010	Limited	1-5	No
Q	Aug. 2010	Limited	1-5	No
R	Aug. 2010	Limited	1-5	No
S	Aug. 2010	Limited	1-5	No
T	May 2008	Limited	1-5	Yes
V	May 2008	Limited	1-5	No
W	May 2008	Limited	1-5	Yes
X	May 2008	Limited	1-5	Yes
Y	May 2008	Limited	1-5	Yes
Z	May 2008	Limited	1-5	No

Based on assigned expertise level, there were five expert measurers and 13 measurers with limited experience.

4.4 Data from experiments

Other than data about experiment participants, there were three types of data collected as outputs of the experimental sessions:

- Defects and issues identified by each participant.
- Effort for each participant.
- Functional size and number of functional processes from measurers.

This thesis chapter is only concerned by the first two types in order to analyse efficiency and effectiveness of adding a measurer to an inspection team. Analysis related to FSM results can be found in chapter 5.

4.4.1 Defects and issues for the whole SRS document

Participants were instructed to read the whole SRS document before focusing their tasks on the functional requirements in section 3.2. However, they were also told that when they identify any defect or issue in other sections than 3.2, they should write it down. Indeed, defects and issues were identified by inspectors and measurers throughout the SRS document.

4.4.1.1 Inspectors defects and issues for the whole SRS document

SRS copies from the 17 inspectors contained 272 defects and 49 issues, for a total of 321 comments, as shown in Table 4.4.

Table 4.4 Number of defects and issues per inspector, by type

		Defects				Issues		
Inspectors	**Type**	**C**	**M**	**S**	**Total**	**I**	**Q**	**Total**
Limited experience	A	4	7	2	**13**	1	1	**2**
	B	3	7	10	**20**	2	--	**2**
	C	3	5	3	**11**	--	2	**2**
	D	3	2	--	**5**	--	2	**2**
	E	1	1	1	**3**	--	--	**0**
	F	--	5	--	**5**	1	--	**1**
	G	5	3	2	**10**	--	--	**0**
	H	7	3	5	**15**	--	--	**0**
	I	5	2	--	**7**	1	--	**1**
	J	--	4	--	**4**	1	--	**1**
	K	2	3	3	**8**	--	--	**0**
	L	--	1	4	**5**	3	2	**5**
	M	18	12	1	**31**	5	6	**11**
	P	7	4	--	**11**	2	--	**2**
Experts	N	20	24	10	**54**	5	1	**6**
	O	12	24	2	**38**	6	--	**6**
	Q	12	11	9	**32**	7	1	**8**
Total:		**102**	**118**	**52**	**272**	**34**	**15**	**49**

Data show a significant difference in the number of critical and minor defects found by expert inspectors and those with limited experience, except for inspector M. Although inspector M pretended to have only one year of experience in document review, the significantly larger number of defects he found suggested that he has been too conservative on his number of years of experience in document review as he already had over 13 years of experience in software development, or that he simply was much more skilled than all other limited experience inspectors.

4.4.1.2 Measurers defects and issues for the whole SRS document

SRS copies from the 18 measurers contained a total of 154 defects and 39 issues, for a total of 193 comments, as shown in Table 4.5. Measurers S and W did not follow the procedure and they did not identify any defect or issue. From this point on, defect data from measurers S and W will be ignored.

Table 4.5 Number of defects and issues per measurer, by type

		Defects				Issues			
Measurers		**C**	**M**	**S**	**Total**	**I**	**Q**	**A**	**Total**
Experts	C	15	12	20	**47**	2	1	--	**3**
	F	5	1	8	**14**	1	1	1	**3**
	H	4	2	5	**11**	--	--	--	**0**
	J	4	2	--	**6**	--	4	5	**9**
	L	10	14	6	**30**	--	1	--	**1**
Limited experience	M	3	--	--	**3**	1	--	--	**1**
	N	3	1	1	**5**	--	--	--	**0**
	O	1	--	--	**1**	--	--	--	**0**
	P	2	--	--	**2**	1	--	6	**7**
	Q	3	1	1	**5**	--	4	7	**11**
	R	3	--	--	**3**	--	1	--	**1**
	S	--	--	--	**0**	--	--	--	**0**
	T	1	4	1	**6**	1	1	--	**2**
	V	5	1	--	**6**	--	--	--	**0**
	W	--	--	--	**0**	--	--	--	**0**
	X	1	2	1	**4**	--	--	--	**0**
	Y	1	--	4	**5**	--	1	--	**1**
	Z	5	1	--	**6**	--	--	--	**0**
	Total	**66**	**41**	**47**	**154**	**6**	**14**	**19**	**39**

Similarly to inspectors, expert measurers found a significantly higher number of critical and minor defects on average (respectively 7.6 critical and 6.2 minor defects/measurer) than the average number of defects found by measurers of limited experience (respectively 2.5 critical and 0.9 minor defects/measurer, excluding measurers S and W).

The 17 inspectors identified individually 272 defects and 49 issues while the 16 measurers who followed the experiment instructions identified 154 defects and 39 issues for a total of 426 defects and 88 issues logged. When adding defects and issues from inspectors and measurers, there was a total of 514 comments logged into the spreadsheet.

4.4.1.3 Uniquely identified defects and issues

As expected, several defects and issues were identified by more than one participant, because the parallel inspection mode was applied. The main outcome of the 'Explain issues (logging meeting)' step was to obtain a list of uniquely identified defects and issues per inspection team, but there was no team per se, only individual inspectors and measurers. Since each defect was identified with a clear location (page, section, paragraph, and line) and the words or elements causing the defect, it became possible for the researcher to identify duplicates, merge them, and identify all participants having found that defect.

In order to analyze efficiency at a later research step, it was required to draw the inventory of uniquely identified defects and issues. The resulting defects and issues log provided all required information to identify those duplicate defects and issues in order to understand what would constitute the totality of all defects and issues found in the uObserve SRS v1.0.

Table 4.6 shows 272 unique defects and 84 unique issues, for a total of 356 unique comments related to the whole SRS document.

Table 4.6 Uniquely identified defects in the whole SRS, sorted by the number of participants having identified those defects and issues

	Defects				**Issues**			
Number of participants	**C**	**M**	**S**	**Total**	**I**	**Q**	**A**	**Total**
1	71	91	34	**196**	36	23	20	**79**
2	22	16	7	**45**	2	2	--	**4**
3	6	2	5	**13**	--	--	--	**--**
4	2	3	4	**9**	--	--	--	**--**
5	--	1	2	**3**	--	--	--	**--**
6	2	1	--	**3**	--	--	--	**--**
7	--	1	--	**1**	--	--	--	**--**
8	2	--	--	**2**	--	--	--	**--**
10	--	--	1	**1**	--	--	--	**--**
Total	**105**	**115**	**53**	**273**	**38**	**25**	**20**	**83**

Over 95% of issues (79/83) were identified by only one participant, which was expected by the researcher as issues are usually participants' personal notes. What was more surprising was that almost 72% (196/273) of defects had been identified by only one participant. It was expected that spelling and syntax defects would have been identified by more than one participant for the majority but it was the case for only 36% of them (19/53).

4.4.1.4 Rejected defects during logging meeting or during defect verification

When inspectors did their individual defect identification task (Step 3), they wrote down comments about any potential defect they perceived. However, some of these potential defects, when explained in Step 4, turned out not to be defects and had to be rejected. As an example, an inspector wrote a critical defect in a specific use case saying that the primary actor was not identified, but the use case clearly stated the actor in the triggering event, as with all other use cases. Therefore this potential defect was rejected as it could not have been confirmed. There was a possibility that this inspector skipped a line when reading.

A total of 30 unique defects and 6 unique issues (only improvement suggestions) were rejected during logging meetings or during defect verification steps:

- Critical defects: 10.
- Minor defects: 18.
- Spelling/syntax: 2.
- Improvement suggestion: 6.

The reasons for rejecting these defects and issues were explained in the defect log and the rejected status was agreed to with the inspection leader. There were three types of defect rejection reasons:

1) The defect identified as critical, minor or spelling was an improvement suggestion that was against customer needs stated in sections 1 and 2 of the SRS document.

2) The identified defect was not a defect as the required information was available in the SRS document, which may have been overlooked by the participant (inspector as well as measurer).
3) The identified defect was asking for design details which were irrelevant for an SRS document.

Questions that were adequately answered had a 'Resolved' status assigned. Other questions could have already been transformed into critical or minor defects or improvement suggestions. One question stayed 'open' as it was not answered but without being considered as a defect.

Table 4.7 shows the resulting unique identified defects and issues, excluding those that were rejected for the reasons previously mentioned.

Table 4.7 Uniquely identified defects in the whole SRS document, excluding rejects

	Defects				**Issues**			
Number of participants	**C**	**M**	**S**	**Total**	**I**	**Q**	**A**	**Total**
1	63	75	32	**170**	31	23	20	**74**
2	20	14	7	**41**	1	2	--	**3**
3	6	2	5	**13**	--	--	--	**--**
4	2	3	4	**9**	--	--	--	**--**
5	--	1	2	**3**	--	--	--	**--**
6	2	1	--	**3**	--	--	--	**--**
7	--	1	--	**1**	--	--	--	**--**
8	2	--	--	**2**	--	--	--	**--**
10	--	--	1	**1**	--	--	--	**--**
Total	**95**	**97**	**51**	**243**	**32**	**25**	**20**	**77**

4.4.2 Defects and issues related to functional requirements

With the instruction of focusing on section 3.2 of the SRS, participants had to read the whole SRS document to understand the content of section 3.2 and they also recorded defects and issues in the rest of the SRS (sections 1, 2, 3.1, and 3.3). Since this research focuses on

functional requirements (FR), defects relevant to section 3.2 of the SRS were isolated. Rejected defects were ignored in order to take into account only the defects confirmed during analysis.

4.4.2.1 Inspectors defects and issues related to FR

Table 4.8 shows defects and issues found by the 17 inspectors that were related to section 3.2 of the SRS.

Table 4.8 Inspectors defects and issues related to Functional Requirements

		Defects				Issues		
Inspectors		**C**	**M**	**S**	**Total**	**I**	**Q**	**Total**
Limited experience	A	4	6	2	**12**	1	1	**2**
	B	3	7	9	**19**	2	0	**2**
	C	3	5	3	**11**	0	1	**1**
	D	3	1	0	**4**	0	2	**2**
	E	1	0	1	**2**	0	0	**0**
	F	0	3	0	**3**	0	0	**0**
	G	2	3	2	**7**	0	0	**0**
	H	5	3	4	**12**	0	0	**0**
	I	5	1	0	**6**	1	0	**1**
	J	0	2	0	**2**	1	0	**1**
	K	2	3	2	**7**	0	0	**0**
	L	0	0	0	**0**	0	0	**0**
	M	12	9	1	**22**	4	4	**8**
	P	6	1	0	**7**	1	0	**1**
Experts	N	6	14	6	**26**	2	1	**3**
	O	9	18	2	**29**	1	0	**1**
	Q	12	11	8	**31**	7	1	**8**
	Total	**73**	**87**	**40**	**200**	**20**	**10**	**30**

4.4.2.2 Measurers defects and issues related to FR

Measurers were instructed to focus on section 3.2 of the SRS, containing the functional requirements. However, whenever a measurer saw defects or issues in the other sections, they

were instructed to identify them as well. Defects and issues on non functional portions of the SRS were logged to be used for updating the SRS at a later research phase. Still, in this section, it is important to isolate defects and issues relevant to functional requirements (FR) and ignoring 'rejected' defects, as shown in Table 4.9.

Table 4.9 Measurers: defects and issues relevant to Functional Requirements

		Defects				Issues			
Measurers	**Type**	**C**	**M**	**S**	**Total**	**I**	**Q**	**A**	**Total**
Experts	C	8	5	6	**19**	0	1	0	**1**
	F	5	1	7	**13**	1	0	1	**2**
	H	4	2	4	**10**	0	0	0	**0**
	J	3	2	0	**5**	0	4	5	**9**
	L	10	14	5	**29**	0	1	0	**1**
Limited experience	M	3	0	0	**3**	0	0	0	**0**
	N	2	0	0	**2**	0	0	0	**0**
	O	1	0	0	**1**	0	0	0	**0**
	P	2	0	0	**2**	1	0	6	**7**
	Q	4	1	1	**6**	0	3	7	**10**
	R	3	0	0	**3**	0	1	0	**1**
	T	1	2	1	**4**	0	1	0	**1**
	V	4	0	0	**4**	0	0	0	**0**
	X	1	2	1	**4**	0	0	0	**0**
	Y	1	0	4	**5**	0	1	0	**1**
	Z	5	1	0	**6**	0	0	0	**0**
	Total	**57**	**30**	**29**	**116**	**2**	**13**	**19**	**33**

4.4.2.3 Uniquely identified defects and issues related to Functional Requirements

Table 4.10 shows 185 unique defects and 61 unique issues, for a total of 246 unique comments related to the Functional Requirements (FR). For efficiency and effectiveness analysis purpose, only critical and minor defects are accounted for: 73 critical + 81 minor = 154 defects relevant to FR.

Table 4.10 Uniquely identified defects and issues related to FR, sorted by the number of measurers having identified those defects and issues

	Defects				Issues			
Number of participants	**C**	**M**	**S**	**Total**	**I**	**Q**	**A**	**Total**
1	45	64	18	**127**	21	20	19	**60**
2	17	10	2	**29**	--	1	--	**1**
3	5	1	5	**11**	--	--	--	**--**
4	2	3	3	**8**	--	--	--	**--**
5	--	1	2	**3**	--	--	--	**--**
6	2	1	--	**3**	--	--	--	**--**
7	--	1	--	**1**	--	--	--	**--**
8	2	--	--	**2**	--	--	--	**--**
10	--	--	1	**1**	--	--	--	**--**
Total	**73**	**81**	**31**	**185**	**21**	**21**	**19**	**61**

4.4.3 Effort data

4.4.3.1 Inspection effort

Effort for identifying defects required not only the individual checking effort but also effort from previous steps and the logging meeting step (Stewart and Priven, 2008). Therefore, inspection effort must include effort to perform steps of the inspection method relevant to the comparison: steps 1 (Plan the inspection) through 4 (Explain issues ‘logging meeting’).

Step 1 (Plan the inspection) was done prior to the experiments and required 15 minutes of effort, mainly to ensure the adequate version of the SRS was to be used and printed, and to prepare and print the inspection form. This effort was spent by the researcher and was in conformity with the inspection approach as applied in the industry. The same effort was required for all three experimental sessions. Therefore, a fixed effort of 15 minutes was accounted for every inspection team.

Step 2 (Open the inspection) consisted of holding an inspection kick-off to describe the context and an overview of the SRS, assign roles to inspectors, and get their commitment to play their assigned role fully. The duration was 10 minutes, which will be accounted for every participant in the analysis section, including inspectors, the researcher, and an acting inspection leader when independent from inspectors. In summary, Step 2 effort represented 10 minutes per inspector plus 20 minutes for the researcher and the inspection leader (e.g. a team of four inspectors had a Step 2 effort of 60 minutes [(4*10)+20=60].

In Step 3 (Inspect the product), effort spent varied with each inspector in the first two experimental sessions. In the third experimental session, the effort was time-boxed to 60 minutes, which was used by all participants.

Step 4 (Explain issues ‘logging meeting’) duration was one hour (60 minutes) for the first experimental session, during which inspectors explained identified defects, focusing on critical and minor defect types. The three inspectors were present the whole time, plus the researcher acting as the SRS author and an independent person acting as the inspection leader.

The duration of Step 4 for the second experimental session was of 2 hours and involved two persons for a total effort of 240 minutes. Those two persons were the researcher acting as the product author in this inspection step, and inspector Q acting as the inspection leader. This second experimental session involved the defect and issues log verification against 10 SRS copies (the physical evidences). Logging meeting effort was driven by the number of defects and issues to verify and discuss. The effort was not split per SRS copy as it was done as a single chuck of time. Instead, to assign a reasonable effort value of this step per SRS copy, the average effort per item (defect or issue) was calculated based on the total number of defects and issues found by each participant of this experiment. Inspectors of this second experiment identified a total of 139 defects and issues with an average of 1.74 minute per defect and issue (240 minutes / 139 defects and issues).

The duration of Step 4 for the third experimental session was of 55 minutes with two persons for a total effort of 110 minutes. Those two persons were the researcher acting as the product author in this inspection step, and an independent person acting as the inspection leader. This third experimental session involved the defect and issues log verification against four SRS copies. The effort was not split per SRS copy as it was done as a single chuck of time. Instead, to assign a reasonable effort value of this step per SRS copy, the average effort per item (defect or issue) was calculated based on the total number of defects and issues found by each participant of this experiment. Inspectors of this third experiment identified a total of 65 defects and issues with an average of 1.69 minute per defect and issue (110 minutes / 65 defects and issues).

Table 4.11 presents a summary of inspection effort per inspector. Effort for Step 4 in the second and third experimental sessions has been calculated with the average values above and the resulting decimal value has been raised to the next minute.

Table 4.11 Summary of inspection effort per inspector, grouped by experiment session

Experimental session	**Inspectors**	**Total defects and issues**	**Effort per step (minutes)**		
			Step 3	**Step 4**	**Total**
First	N	60	55	60	115
	O	44	55	60	115
	P	13	60	60	120
Second	A	15	45	27	72
	B	22	60	39	99
	C	13	50	23	73
	D	7	75	13	88
	E	3	75	6	81
	F	6	90	11	101
	G	10	60	18	78
	H	5	60	9	69
	I	8	60	14	74
	Q	40	65	70	135
Third	J	5	60	9	69
	K	8	60	14	74
	L	10	60	17	77
	M	42	60	71	131

In addition, Step 1 was calculated as a total of 15 minutes per inspection team, regardless of the number of participants. Step 2 accounted for 10 minutes per participant plus 10 minutes for the acting author plus 10 minutes for the independent inspection leader (first and third experiment). These values were added for data analysis.

4.4.3.2 Measurement effort

Measurers participated in Step 2 of the inspection approach where the SRS context was explained and definitions for defects and rules were given. Therefore, an additional 10 minutes per measurer was added for analysis purpose.

Also, defect verification effort must be accounted for the same way it was calculated for inspectors. This effort varied by experimental session and has to be averaged by the total number of defects and issues found by measurers (Table 4.12). Total effort to explain and verify defects and issues was 190 minutes (90+60+40) for 193 items (109+34+50).

Table 4.12 Measurers effort to explain and verify defects and issues

Effort calculation elements	**Experimental session**		
	First	**Second**	**Third**
Verification duration (minutes)	15	30	20
# of participants	6	2	2
Verification effort (minutes)	**90**	**60**	**40**
# items (defects + issues)	109	34	50
Average effort per item	0.83	2.00	0.74
Measurers involved	C, F, H, L	T, V, W, X, Y, Z	J, M, N, O, P, Q, R, S

Table 4.13 shows the effort (in minutes) spent by each measurer. This effort includes the measurement and defect identification activity that was done in parallel of inspectors' Step 3, plus the 10 minutes kick-off and the calculated verification effort from average effort per item (defects + issues).

Table 4.13 Effort spent by measurers, including all steps they were involved in

Exp. session	Measurers	Total defects and issues	Kick-off effort	Measurement effort (minutes)	Verification effort (minutes)	Total effort
First	C	50	10	75	42	127
	F	17	10	49	14	73
	H	11	10	45	9	64
	L	31	10	60	26	96
Second	T	8	10	49	16	75
	V	6	10	65 [1]	12	87
	X	4	10	80	8	98
	Y	6	10	90	12	112
	Z	6	10	65	12	87
Third	J	15	10	60	11	81
	M	4	10	60	3	73
	N	5	10	60	4	74
	O	1	10	60	1	71
	P	9	10	60	7	77
	Q	16	10	60	12	82
	R	4	10	60	3	73
Total		**193**	**160**	**998**	**192** [2]	**1350**
Average		**12.1**	**10.0**	**62.4**	**12.0**	**84.4**
Std dev.		**12.1**	**0.0**	**11.0**	**9.7**	**16.1**

Note [1]: Measurer V did not write his effort on the form but he had given his SRS copy and measurement results at the same time as measurer Z, +/- 2 minutes.
Note [2]: The difference on the total was due to rounding error.

Data from measurers S and W were excluded since they did not follow the experimental procedure.

4.5 Analysis of efficiency and unit cost

4.5.1 Assumptions for analysis

4.5.1.1 Assumptions related to derived measures

To analyze data related to the first three research objectives the following derived measures are required:

- Efficiency = number of defects found during inspections / total number of defects found [during the development project]. When the uObserve system was developed in 2004, the lead developer reported only one residual defect at delivery that was traced back to the design, not to the requirements. Defects during development were not recorded. Therefore, this research project made the assumption that 100% of uniquely identified critical and minor defects related to FR constituted the 'total number of defects found'; in this research project, this value corresponds to 73 critical defects plus 81 minor defects, for a total of 154 unique defects (Table 4.10).

- Unit cost (effectiveness) = review effort / number of defects identified. For the purpose of this research project, an assumption was made that only defects of critical and minor types were accounted for. Defects of spelling/syntax type were ignored, mainly because industry defect data report only on major (critical) and minor defect types (Weigers, 2002, pp. 129-134). Effort data must include effort from the first four steps (plan, open, inspect, and explain). The unit cost (effectiveness) derived measure is referred as 'Effort.per.Defect' by Weigers.

It has been decided to assign 100% of effort to FR only for analysis purposes.

4.5.1.2 Assumptions related to the number of participants in an inspection team

To analyze inspection efficiency and effectiveness, only unique critical and minor defects must be considered for any team of inspectors. Removing duplicates defects assumed a situation where a defined number of specific inspectors would be part of the same inspection team and that duplicates among those team members were identified and removed in order to obtain the number of unique critical and minor defects that the team found. As the industry usually selects two to four inspectors per inspection team of an SRS document, simulations with these sizes of inspection teams was done: two, three, and four team members.

Since this research is designed to investigate the contribution of adding one measurer to an inspection team, five active inspectors (including the measurer) in a team could be considered too large and too expensive. For analysis purposes, inspection teams of two to four inspectors plus one measurer will be selected. It could be desirable to analyze all possible combinations of inspectors to form inspection teams. For practical reasons, the analyses were made on the two extreme cases and on the median case – for the three team sizes of two, three, and four inspectors – grouping inspectors together based on the number of unique critical and minor defects each team member have found.

The first extreme case could be called the ‘best’ team of inspectors with those inspectors who identified the highest number of critical and minor defects. The second extreme case could be called the ‘worst’ team of inspectors with those inspectors who identified the lowest number of critical and minor defects. The ‘median’ team of inspectors represented inspectors whose number of critical and minor defects found ended in the median ranges of the remaining inspectors, i.e. inspectors who were not part of the worst or the best teams. Table 4.14 provides the total number of critical and minor defects related to FR of each inspector, sorted in ascending order.

Table 4.14 Critical and minor defects related to FR, per inspector

Experience	Inspector defects				Average defects
	Inspectors	**C**	**M**	**Total**	
Limited	L	0	0	**0**	**5.3**
	E	1	0	**1**	
	J	0	2	**2**	
	F	0	3	**3**	
	D	3	1	**4**	
	K	2	3	**5**	
	G	2	3	**5**	
	I	5	1	**6**	
	P	6	1	**7**	
	C	3	5	**8**	
	H	5	3	**8**	
	B	3	7	**10**	
	A	4	6	**10**	
Expert	N	6	14	**20**	**22.8**
	M	12	9	**21**	
	Q	12	11	**23**	
	O	9	18	**27**	
	Total			**160**	
	Average			**9.4**	
	Standard deviation			**8.0**	
	Median			**7.0**	

Inspector L did find defects but he did not find critical or minor defects in the FR section of the SRS; he was still kept in the data to be analyzed as he followed the procedure. Measurers with limited experience found 5.3 defects on average, compared to 22.8 defects on average found by experts. The average number of defects found by experts increases the average number of defects found by inspectors with limited experience by 330% ((22.8-5.3)/5.3).

4.5.1.3 Assumption of FSM without inspection

Inspections, or any other form of peer reviews, are practices defined at CMMI level 3. An organization may not have integrated this practice into its software process when its maturity or capacity level might not be at CMMI level 3. Taking this into consideration, it is relevant

to analyze defects data found by measurers even when inspections are not done at all in order to understand and quantify the efficiency and effectiveness of measurers in finding defects.

4.5.2 Regrouping inspectors into inspection teams for analysis

4.5.2.1 The best teams of inspectors

In Table 4.14, the four inspectors with the highest number of critical and minor defects were inspectors O, Q, M, and N. Inspector M was considered of limited experience and inspectors N, O, and Q were considered experts. Assembling team members for analysis provided the following team compositions:

- Best-4: all four inspectors O, Q, M, and N.
- Best-3: the three inspectors with the highest number of defects, O. Q, and M.
- Best-2: the two inspectors with the highest number of defects, O and Q.

Extracting defects data from the defects log for each of these best teams of inspection provided the results in Table 4.15. Each inspector found a certain number of defects that were added. Then, duplicates defects had to be identified and subtracted from the total to obtain the number of unique functional defects found. Some duplicate defects may have been found by two or three of the inspectors. When a defect was found by two inspectors, that defect was counted twice in the total and one of the duplicates needed to be subtracted. When a defect was found by three inspectors, that defect was counted three times in the total and two of the duplicates needed to be subtracted. To calculate the number of duplicates, the following formula was used:

$$\text{Duplicates} = \text{number of defects found} \times (\text{number of inspectors} - 1) \qquad (4.1)$$

Table 4.15 Defects found by each best team of inspectors and corresponding efficiency

Best teams of inspectors	Best-4	Best-3	Best-2
Inspectors	O, Q, M, and N	O. Q, and M	O and Q
Total number of defects found by each team member	**91**	**71**	**50**
Duplicates: - Defects found by 2 inspectors - Defects found by 3 inspectors	 8 1 x (3-1) = 2	 3 1 x (3-1) = 2	 2 --
Minus: total number of duplicate defects:	10	5	2
Number of unique functional defects found	**81**	**66**	**48**
Efficiency (based on 154 unique defects)	**52.6%**	**42.9%**	**31.2%**

As expected, more inspectors in an inspection team increases the efficiency as additional defects are identified by the additional inspector.

The calculated effort for each of the best teams of inspectors is presented in Table 4.16.

Table 4.16 Effort spent by each best team of inspectors and corresponding unit cost

Best teams of inspectors	Best-4	Best-3	Best-2
Inspectors	O, Q, M, and N	O. Q, and M	O and Q
Number of unique functional defects found	**81**	**66**	**48**
Effort for Step 1 (fixed)	15	15	15
Effort for Step 2	60	50	40
Effort for Step 3 & 4 (see Table 4.11)	496	381	250
Total inspection effort	**571**	**446**	**305**
Unit cost (in minutes/defect)	**7.0**	**6.8**	**6.4**

From Table 4.16 it can be observed that while adding an inspector, additional unique defects are found but it also increased the total inspection effort of the inspection team. The bottom line of Table 4.16 indicates therefore that the additional effort (i.e. adding an inspector) outweighs the additional number of defects found: the unit cost increases, i.e. from an average effort of 6.4 minutes per defect for the Best-2 team up to 7.0 minutes per defect for the Best-4 team of inspectors. However, the increase is minor and could be easily offset by the greater number of defects identified and corrected, leading to less rework later in the development life cycle.

4.5.2.2 The median teams of inspectors

In Table 4.14, inspectors around the median value of defects found by inspector (7 defects/inspector, as found by inspector P) are selected. To assemble the median teams of inspectors, inspector P is selected, adding the inspector after P (inspector C), then the one before P (inspector I), and finally the next inspector after inspector C (inspector H). Assembling team members for analysis provided the following team compositions:

- Median-4: inspectors I, P, C, and H.
- Median-3: inspectors I, P, and C.
- Median-2: inspectors P and C.

Extracting defects data from the defects log for each of these median teams of inspection provided the results in Table 4.17.

Table 4.17 Defects found by each median team of inspectors & corresponding efficiency

Median teams of inspectors	Median-4	Median-3	Median-2
Inspectors	I, P, C, and H	I, P, and C	P and C
Total number of defects found by each team member	**29**	**21**	**15**
Minus: total number of duplicate defects	3	1	0
Number of unique functional defects found	**26**	**20**	**15**
Efficiency (based on 154 unique defects)	**16.9%**	**13.0%**	**9.7%**

The efficiency of median teams of inspectors increased as the number of team members increased.

The calculated effort for each of the Median teams is in Table 4.18.

Table 4.18 Effort spent by each Median team of inspectors & corresponding unit cost

Median teams of inspectors	Median-4	Median-3	Median-2
Inspectors	I, P, C, and H	I, P, and C	P and C
Number of unique functional defects found	**26**	**20**	**15**
Effort for Step 1 (fixed)	15	15	15
Effort for Step 2	60	50	40
Effort for Step 3 & 4 (see Table 4.11)	336	267	193
Total inspection effort	**411**	**332**	**248**
Unit cost (in minutes/defect)	**15.8**	**16.6**	**16.5**

Unit cost had only 0.1 minute/defect between the Median-2 and the Median-3 teams, and the difference was less than one minute per defect (16.6-15.8=0.8) between the Median-3 and the Median-4 teams. These results are considered stable.

4.5.2.3 The worst teams of inspectors

In Table 4.14, the four inspectors with the lowest number of critical and minor defects are inspectors L, E, J, and F: these inspectors are considered having limited experience. Assembling team members for analysis provided the following team compositions:

- Worst-4: all four inspectors L, E, J, and F.
- Worst-3: the three inspectors with the lowest number of defects, L, E, and J.
- Worst-2: the two inspectors with the lowest number of defects, L and E.

Extracting defect data from the defects log for each of these worst teams of inspectors provided the results in Table 4.19.

Table 4.19 Defects found by each worst team of inspectors & corresponding efficiency

Worst teams of inspectors	Worst-4	Worst-3	Worst-2
Inspectors	L, E, J, and F	L, E, and J	L and E
Total number of defects found by each team member	**6**	**3**	**1**
Minus: total number of duplicate defects	--	--	--
Number of unique functional defects found	**6**	**3**	**1**
Efficiency (based on 154 unique defects)	**3.9%**	**1.9%**	**0.6%**

The efficiency of worst teams of inspectors increased as the number of team members increased.

The calculated effort for each of the worst teams is in Table 4.20.

Table 4.20 Effort spent by each worst team of inspectors & corresponding unit cost

Worst teams of inspectors	Worst-4	Worst-3	Worst-2
Inspectors	L, E, J, and F	L, E, and J	L and E
Number of unique functional defects found	**6**	**3**	**1**
Effort for Step 1 (fixed)	15	15	15
Effort for Step 2	60	50	40
Effort for Step 3 & 4 (see Table 4.11)	328	227	158
Total inspection effort	**403**	**292**	**213**
Unit cost (in minutes/defect)	**67.2**	**97.3**	**213.0**

There is a large variation among unit cost of worst teams of inspectors, ranging from 213.0 minutes/defects to 67.2 minutes/defect. The only defect found by the Worst-2 team was tripled when a third member was added, then doubled when a fourth member was added. The small number of defects found was responsible for the relatively high unit cost.

4.5.2.4 Impact on efficiency of adding one inspector

Efficiency is calculated as the percentage of critical and minor defects found by an inspection team when compared to the total number of critical and minor defects found by all

participants. An efficiency value of 100% would mean that all defects were found by a single team. A higher efficiency value represents a better performance.

Whether an inspection team has two, three, or four inspectors impact the number of defects this team would find (Figure 4.2). As the number of inspectors increases, the efficiency increases also:

- The Worst teams increased their efficiency from 0.6% to 1.9% (a 200% increase) and from 1.9% to 3.9% (a 100% increase) with each additional team member.
- The Median teams increased their efficiency from 9.7% to 13.0% (a 33% increase) and from 13.0% to 16.9% (a 30% increase) with each additional team member.
- The Best teams increased their efficiency from 31.2% to 42.9% (a 38% increase) and from 42.9% to 52.6% (a 23% increase) by adding the third and the fourth inspector having found the highest number of defects.

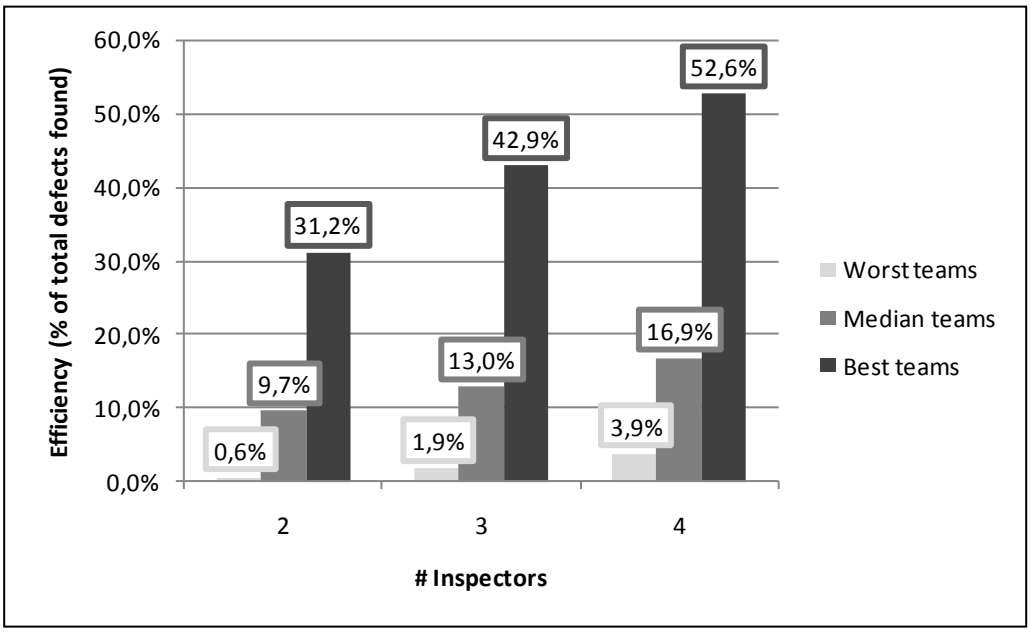

Figure 4.2 Efficiency of inspection teams, grouped by number of inspectors in each team

Efficiency results show significant differences among the three groups of teams, suggesting that review results were indeed related to reviewers' skills or experience. Based on these

results, an organization would increase inspections' efficiency with several expert inspectors in their inspection teams.

4.5.2.5 Impact on unit cost of adding one inspector

Unit cost is calculated as the average number of minutes per defect. The lower the unit cost value is the better for effectiveness.

The Worst teams of inspectors improve significantly their unit cost as the number of inspectors increases: the average effort per defect goes down from 213.0 minutes with two inspectors to 67.2 minutes with four inspectors. The unit cost of Median teams is 16.5 minutes/defects with two inspectors and varies to 16.6 and 15.8 with the addition of each inspector. The Best teams of inspectors have an average unit cost of 6.7 minutes/defect with a standard deviation of 0.2. However, as new members are added to the Best teams, the unit cost goes up (Figure 4.3).

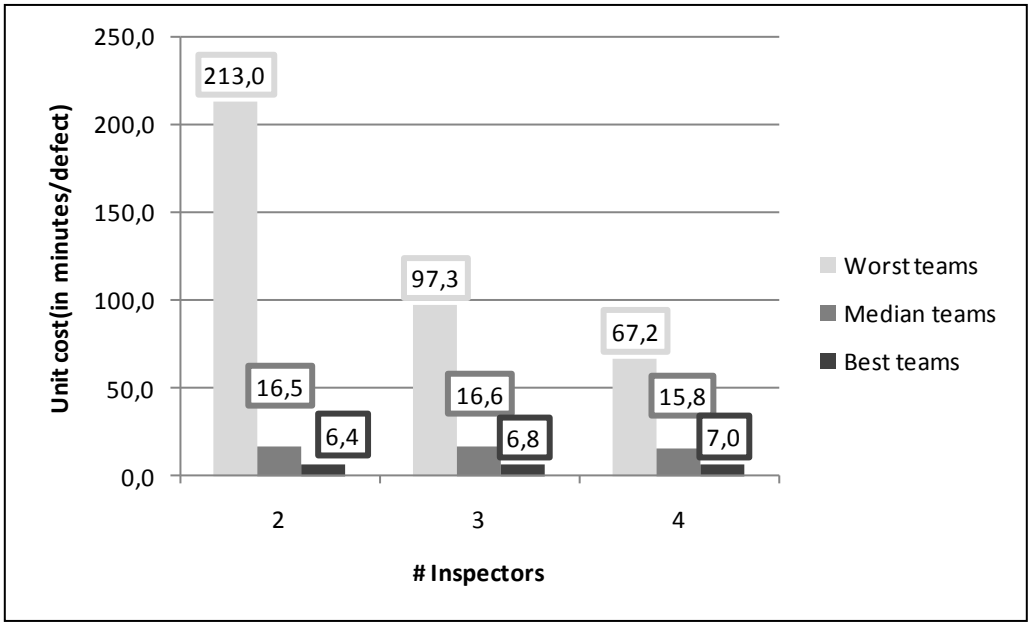

Figure 4.3 Unit cost of inspection teams, grouped by number of inspectors in each team

The unit cost variations are less than one minute/defect among Best or Median teams of two, three, and four inspectors. The increasing number of defects found as one inspector is added to each team may add a greater value than the effort related cost of fixing those defects at a later life-cycle phase.

Unit cost results show differences among the three groups of teams, suggesting that review results are indeed related to reviewers' skills or experience. Based on these results, an organization would improve inspections' unit cost with several expert inspectors in their inspection teams.

4.5.3 Efficiency analysis of measurers in finding defects

In industry practice, the likelihood of having more than one measurer per SRS document is low. For any given software project, an assumption is made that there would be only one measurer measuring the functional size in a project team. Therefore, defects found by measurers are kept and analyzed individually in the analyses presented here.

As shown in Table 4.15, Table 4.17, and Table 4.19, every added member to an inspection team contributed to increase the team's efficiency. Because all measurers found defects (Table 4.9), it was expected that adding a measurer to an inspection team would also increase the team's efficiency. However, adding either an inspector or a measurer to an inspection team may not increase the efficiency with the same results. To understand which of an extra inspector or of adding a measurer to an inspection team provides the higher efficiency increase, simulations are done as follows:

1) Add an extra inspector to each team, selected based on rank (worst, median, or best).
2) Remove duplicate defects to obtain the increased number of unique defects found by the team members including the extra inspector.
3) Calculate the increased efficiency with extra inspector as the increased number of unique defects found divided by 154 defects (100% of defects found, value shown as a percentage).

4) Calculate the extra inspector's efficiency increase as the difference between the increased efficiency and the inspection team's efficiency before adding the extra inspector. Negative values are not possible.
5) Add each measurer to each inspection team (without the extra inspector).
6) Remove duplicate defects to obtain the increased number of unique defects found by the team members including the measurer.
7) Calculate the increased efficiency with a measurer as the increased number of unique defects found divided by 154 defects (100% of defects found, value shown as a percentage).
8) Calculate the measurer's efficiency improvement as the difference between the increased efficiency and the inspection team's efficiency before adding the measurer. Negative values are not possible.
9) Calculate the net gain (or loss) as the difference between the measurer's efficiency improvement and the extra inspector's efficiency increase. A positive value means that the measurer increased the efficiency with a higher value than adding an extra inspector. A negative value is called a net loss which means that the measurer increased the efficiency with a lower value than adding an extra inspector.
10) Calculate the average net gain (or loss) and standard deviation on net gain (or loss) among measurers.

To analyze and understand the potential efficiency improvement of each measurer over an inspection team, any defect that is also found by at least one inspector of a given team had to be removed from the defects found by each measurer (the duplicate defects). Therefore, the efficiency of adding a measurer over an inspection team is calculated with measurers' added unique defects only. The minimum potential efficiency improvement would be 0% as measurers could not come up with a negative value of added defects. Therefore, any single unique defect found by a measurer is improving the inspection team efficiency but may not lead to a positive net gain. To obtain a positive net gain, a measurer has to find at least the same number of defects as the selected extra inspector.

The following subsections analyse the measurers' defects data using inspection teams of four, three and two inspectors. All tables show the number of critical and minor defects found by measurers, the number of duplicate defects each measurer has with each inspection team, the number of defects each measurer is adding to each inspection team, the new efficiency of the inspection team with the added measurer, the efficiency improvement over that inspection team without a measurer, and the net gain (or loss if the value is negative) of efficiency (i.e. when a measurer is replacing an extra inspector).

4.5.3.1 Measurers as a fifth member of an inspection team

Worst-4 team

When adding the next worst inspector (inspector D) to Worst-4 team and removing duplicate defects, the number of defects found goes up from 6 to 9. The efficiency was 3.9% (6/154) and its new value is 5.8% (9/154). The efficiency increase for adding a fifth inspector to Worst-4 inspection team is 1.9% (see top portion of Table 4.21).

When considering the Worst-4 team, the average efficiency improvement for adding measurers goes up from 3.9% to 7.2% , which correspond to an efficiency increase of 3.3% which exceeds the 1.9% increased efficiency of going from four to five inspectors. The average net gain is 1.4% (right-hand side column in Table 4.21). All measurers improved the efficiency of the Worst-4 inspection team (column 'Improvement' in Table 4.21). In addition, eleven measurers out of 16, including all experts (C, F, H, J, and L), had a positive net gain of efficiency over adding a fifth inspector to the Worst-4 inspection team (right-hand side column in Table 4.21, 'Net gain or loss (-)').

Table 4.21 Efficiency improvement when adding a measurer over the Worst-4 inspection team

Worst-4						
With 5th inspector D			Inspectors L, E, J, and F			Increase from 5th inspector
Defects	Efficiency		Defects	Efficiency		
9	5.8%		6	3.9%		1.9%
Measurer	C&M defects	Dupl.	Added	Effici-ency	Impro-vement	Net gain or loss(-)
C	13	1	12	11.7%	7.8%	5.9%
F	6	1	5	7.1%	3.2%	1.3%
H	6	1	5	7.1%	3.2%	1.3%
J	5	0	5	7.1%	3.2%	1.3%
L	24	0	24	19.5%	15.6%	13.7%
M	3	0	3	5.8%	1.9%	0.0%
N	2	0	2	5.2%	1.3%	-0.6%
O	1	0	1	4.5%	0.6%	-1.3%
P	2	0	2	5.2%	1.3%	-0.6%
Q	5	0	5	7.1%	3.2%	1.3%
R	3	0	3	5.8%	1.9%	0.0%
T	3	1	2	5.2%	1.3%	-0.6%
V	4	0	4	6.5%	2.6%	0.7%
X	3	0	3	5.8%	1.9%	0.0%
Y	1	0	1	4.5%	0.6%	-1.3%
Z	6	0	6	7.8%	3.9%	2.0%
Total	**87**	**4**	**83**	**--**	**--**	**--**
Average	**5.4**	**0.3**	**5.2**	**7.2%**	**3.3%**	**1.4%**
Std dev	**5.5**	**0.4**	**5.5**	**3.6%**	**3.6%**	**3.6%**
Min	**1**	**0**	**1**	**4.5%**	**0.6%**	**-1.3%**
Max	**24**	**1**	**24**	**19.5%**	**15.6%**	**13.7%**

The average number of added defects found by measurers (5.2) compared to the number of defects found by the four inspectors of the Worst-4 team (6), represents a 86.5% increase (5.2/6).

Median-4 team

When adding the next median inspector (inspector G) to Median-4 team and removing duplicate defects, the number of defects found goes up from 26 to 28. The efficiency was 16.9% (26/154) and its new value is 18.2% (28/154). The efficiency increase for adding a fifth inspector to Median-4 inspection team is 1.3% (see top portion of Table 4.22).

When considering the Median-4 team, the average efficiency improvement for adding measurers (3.2%) also exceeds their efficiency of going from four to five inspectors, which was of 1.3%. The average net gain is 1.9% (Table 4.22). All measurers – except measurer Y – improve the efficiency of the Median-4 inspection team. Fourteen measurers out of 16, including all experts, had a positive net gain of efficiency over adding a fifth inspector to the Median-4 inspection team.

Table 4.22 Efficiency improvement when adding a measurer over the Median-4 inspection team

Median-4						
With 5th inspector G			Inspectors P, C, H, and I			Increase from 5th inspector
Defects	Efficiency		Defects	Efficiency		
28	18.2%		26	16.9%		1.3%
Measurer	C&M defects	Dupl.	Added	Effici-ency	Impro-vement	Net gain or loss(-)
C	13	1	12	24.7%	7.8%	6.5%
F	6	0	6	20.8%	3.9%	2.6%
H	6	0	6	20.8%	3.9%	2.6%
J	5	2	3	18.8%	1.9%	0.6%
L	24	1	23	31.8%	14.9%	13.6%
M	3	0	3	18.8%	1.9%	0.6%
N	2	0	2	18.2%	1.3%	0.0%
O	1	0	1	17.5%	0.6%	-0.7%
P	2	0	2	18.2%	1.3%	0.0%
Q	5	0	5	20.1%	3.2%	1.9%
R	3	0	3	18.8%	1.9%	0.6%
T	3	1	2	18.2%	1.3%	0.0%
V	4	1	3	18.8%	1.9%	0.6%
X	3	1	2	18.2%	1.3%	0.0%
Y	1	1	0	16.9%	0.0%	-1.3%
Z	6	0	6	20.8%	3.9%	2.6%
Total	**87**	**8**	**79**	**--**	**--**	**--**
Average	**5.4**	**0.5**	**4.9**	**20.1%**	**3.2%**	**1.9%**
Std dev	**5.5**	**0.6**	**5.4**	**3.5%**	**3.5%**	**3.5%**
Min	**1**	**0**	**0**	**16.9%**	**0.0%**	**-1.3%**
Max	**24**	**2**	**23**	**31.8%**	**14.9%**	**13.6%**

The average number of added defects found by measurers (4.9) compared to the number of defects found by the four inspectors of the Median-4 team (26), represents a 19.0% increase (4.9/26).

Best-4 team

When adding the next best inspector (inspector A) to Best-4 team and removing duplicate defects, the number of defects found goes up from 81 to 88. The Best-4 inspection team efficiency is 52.6% (81/154) and adding an inspector brings its efficiency to 57.1% (88/154), an improvement of 4.5% (see top portion of Table 4.23).

When considering the Best-4 team, the average increase in efficiency for adding measurers (2.7%) is below their efficiency of going from four to five inspectors, which was of 4.5%. The net loss is -1.8% (Table 4.23). Two measurers among experts had a positive net gain of efficiency over adding a fifth inspector to the Best-4 inspection team.

Table 4.23 Efficiency improvement when adding a measurer over the Best-4 inspection team

Best-4						
With 5th inspector A			Inspectors O, Q, M, and N			Increase from 5th inspector
Defects	Efficiency		Defects	Efficiency		
88	57.1%		81	52.6%		4.5%
Measurer	C&M defects	Dupl.	Added	Effici-ency	Impro-vement	Net gain or loss(-)
C	13	5	8	57.8%	5.2%	0.7%
F	6	2	4	55.2%	2.6%	-1.9%
H	6	1	5	55.8%	3.2%	-1.3%
J	5	1	4	55.2%	2.6%	-1.9%
L	24	5	19	64.9%	12.3%	7.8%
M	3	1	2	53.9%	1.3%	-3.2%
N	2	1	1	53.2%	0.6%	-3.9%
O	1	0	1	53.2%	0.6%	-3.9%
P	2	0	2	53.9%	1.3%	-3.2%
Q	5	0	5	55.8%	3.2%	-1.3%
R	3	0	3	54.5%	1.9%	-2.6%
T	3	2	1	53.2%	0.6%	-3.9%
V	4	0	4	55.2%	2.6%	-1.9%
X	3	1	2	53.9%	1.3%	-3.2%
Y	1	0	1	53.2%	0.6%	-3.9%
Z	6	0	6	56.5%	3.9%	-0.6%
Total	**87**	**19**	**68**	**--**	**--**	**--**
Average	**5.4**	**1.2**	**4.3**	**55.3%**	**2.7%**	**-1.8%**
Std dev	**5.5**	**1.6**	**4.3**	**2.8%**	**2.8%**	**2.8%**
Min	**1**	**0**	**1**	**53.2%**	**0.6%**	**-3.9%**
Max	**24**	**5**	**19**	**64.9%**	**12.3%**	**7.8%**

The average number of added defects found by measurers (4.3) compared to the number of defects found by the four inspectors of the Best-4 team (81), represents a 5.2% increase (4.3/81).

Provided functional size

All measurers provide a functional size of the set of requirements within the same time-frame than the inspectors take to only identify defects. All measurers – to the exception of measurer Y with the Median-4 team – identified new defects that each team of four inspectors did not identify.

4.5.3.2 Measurers as a fourth member of an inspection team

Worst-3 team

Adding the next worst inspector (inspector F) to the Worst-3 inspection team is the same as considering the Worst-4 inspection team. The Worst-4 inspection team efficiency is 3.9% (6/154) and the Worst-3 inspection team efficiency is 1.9% (3/154). The efficiency improvement for adding a fourth inspector to Worst-3 inspection team is 2.0% (see top portion of Table 4.24).

When considering the Worst-3 inspection team, the average efficiency improvement for adding a measurer (3.4%) exceeds the efficiency increase of going from three to four inspectors, which is 2.0%. The net gain is 1.4% (Table 4.24). All measurers improved the efficiency of the Worst-3 inspection team. Eleven measurers out of 16, including all experts, have a positive net gain of efficiency over adding a fourth inspector to the Worst-3 inspection team.

Table 4.24 Efficiency improvement when adding a measurer over the Worst-3 inspection team

Worst-3						
With 4th inspector F			Inspectors L, E, and J			Increase from 4th inspector
Defects	Efficiency		Defects	Efficiency		
6	3.9%		3	1.9%		2.0%
Measurer	C&M defects	Dupl.	Added	Effici-ency	Impro-vement	Net gain or loss(-)
C	13	1	12	9.7%	7.8%	5.8%
F	6	1	5	5.2%	3.3%	1.3%
H	6	1	5	5.2%	3.3%	1.3%
J	5	0	5	5.2%	3.3%	1.3%
L	24	0	24	17.5%	15.6%	13.6%
M	3	0	3	3.9%	2.0%	0.0%
N	2	0	2	3.2%	1.3%	-0.7%
O	1	0	1	2.6%	0.7%	-1.3%
P	2	0	2	3.2%	1.3%	-0.7%
Q	5	0	5	5.2%	3.3%	1.3%
R	3	0	3	3.9%	2.0%	0.0%
T	3	1	2	3.2%	1.3%	-0.7%
V	4	0	4	4.5%	2.6%	0.6%
X	3	0	3	3.9%	2.0%	0.0%
Y	1	0	1	2.6%	0.7%	-1.3%
Z	6	0	6	5.8%	3.9%	1.9%
Total	**87**	**4**	**83**	**--**	**--**	**--**
Average	**5.4**	**0.3**	**5.2**	**5.3%**	**3.4%**	**1.4%**
Std dev	**5.5**	**0.4**	**5.5**	**3.6%**	**3.6%**	**3.6%**
Min	**1**	**0**	**1**	**2.6%**	**0.7%**	**-1.3%**
Max	**24**	**1**	**24**	**17.5%**	**15.6%**	**13.6%**

The average number of added defects found by measurers (5.2) compared to the number of defects found by the three inspectors of the Worst-3 team (3), represents a 173.3% increase (5.2/3).

Median-3 team

Adding the next worst inspector (inspector I) to the Median-3 inspection team is the same as considering the Median-4 inspection team. The Median-4 inspection team efficiency is 16.9% (26/154) and the Median-3 inspection team efficiency is 13.0% (20/154), providing a net gain of 3.9% when adding a fourth inspector to the Median-3 inspection team (see top portion of Table 4.25).

The average efficiency improvement for adding a measurer to the Median-3 team (3.3%) is below when a fourth inspector would have been added (3.9%), with a net loss of -0.6% (Table 4.25). All measurers improved the efficiency of the Median-3 inspection team but only five had a positive net gain.

Table 4.25 Efficiency improvement when adding a measurer over the Median-3 inspection team

Median-3						
With 4th inspector I			Inspectors P, C, and H			Increase from 4th inspector
Defects	Efficiency		Defects	Efficiency		
26	16.9%		20	13.0%		3.9%
Measurer	C&M defects	Dupl.	Added	Effici-ency	Impro-vement	Net gain or loss(-)
C	13	0	13	21.4%	8.4%	4.5%
F	6	0	6	16.9%	3.9%	0.0%
H	6	0	6	16.9%	3.9%	0.0%
J	5	2	3	14.9%	1.9%	-2.0%
L	24	1	23	27.9%	14.9%	11.0%
M	3	0	3	14.9%	1.9%	-2.0%
N	2	0	2	14.3%	1.3%	-2.6%
O	1	0	1	13.6%	0.6%	-3.3%
P	2	0	2	14.3%	1.3%	-2.6%
Q	5	0	5	16.2%	3.2%	-0.7%
R	3	0	3	14.9%	1.9%	-2.0%
T	3	0	3	14.9%	1.9%	-2.0%
V	4	1	3	14.9%	1.9%	-2.0%
X	3	1	2	14.3%	1.3%	-2.6%
Y	1	1	0	13.0%	0.0%	-3.9%
Z	6	0	6	16.9%	3.9%	0.0%
Total	**87**	**6**	**81**	**--**	**--**	**--**
Average	**5.4**	**0.4**	**5.1**	**16.3%**	**3.3%**	**-0.6%**
Std dev	**5.5**	**0.6**	**5.5**	**3.5%**	**3.5%**	**3.5%**
Min	**1**	**0**	**0**	**13.0%**	**0.0%**	**-3.9%**
Max	**24**	**2**	**23**	**27.9%**	**14.9%**	**11.0%**

The average number of added defects found by measurers (5.1) compared to the number of defects found by the three inspectors of the Median-3 team (20), represents a 25.5% increase (5.1/20).

Best-3 team

Adding the next best inspector (inspector N) to the Best-3 inspection team is the same as considering the Best-4 inspection team. The Best-4 inspection team efficiency is 52.6% (81/154) and the Best-3 inspection team efficiency is 42.9% (66/154). The efficiency improvement for adding a fourth inspector to Best-3 inspection team is 9.7% (see top portion of Table 4.26).

The average efficiency improvement for adding a measurer to the Best-3 team (3.0%) is below the efficiency increase of adding a fourth inspector (9.7%). Only one expert measurer had a positive net gain over the Best-3 inspection team and it was below the efficiency improvement of adding a fourth inspector (Table 4.26).

Table 4.26 Efficiency improvement when adding a measurer over the Best-3 inspection team

Best-3						
With 4th inspector N			Inspectors O, Q, and M			Increase from 4th inspector
Defects	Efficiency		Defects	Efficiency		
81	52.6%		66	42.9%		9.7%
Measurer	C&M defects	Dupl.	Added	Effici-ency	Impro-vement	Net gain or loss(-)
C	13	4	9	48.7%	5.8%	-3.9%
F	6	2	4	45.5%	2.6%	-7.1%
H	6	1	5	46.1%	3.2%	-6.5%
J	5	1	4	45.5%	2.6%	-7.1%
L	24	3	21	56.5%	13.6%	3.9%
M	3	0	3	44.8%	1.9%	-7.8%
N	2	0	2	44.2%	1.3%	-8.4%
O	1	0	1	43.5%	0.6%	-9.1%
P	2	0	2	44.2%	1.3%	-8.4%
Q	5	0	5	46.1%	3.2%	-6.5%
R	3	0	3	44.8%	1.9%	-7.8%
T	3	2	1	43.5%	0.6%	-9.1%
V	4	0	4	45.5%	2.6%	-7.1%
X	3	0	3	44.8%	1.9%	-7.8%
Y	1	0	1	43.5%	0.6%	-9.1%
Z	6	0	6	46.8%	3.9%	-5.8%
Total	**87**	**13**	**74**	**--**	**--**	**--**
Average	**5.4**	**0.8**	**4.6**	**45.9%**	**3.0%**	**-6.7%**
Std dev	**5.5**	**1.2**	**4.7**	**3.0%**	**3.0%**	**3.0%**
Min	**1**	**0**	**1**	**43.5%**	**0.6%**	**-9.1%**
Max	**24**	**4**	**21**	**56.5%**	**13.6%**	**3.9%**

The average number of added defects found by measurers (4.6) compared to the number of defects found by the three inspectors of the Best-3 team (66), represents a 7.0% increase (4.6/66).

4.5.3.3 Measurers as a third member of an inspection team

Worst-2 team

Adding the next worst inspector (inspector J) to the Worst-2 inspection team is the same as considering the Worst-3 inspection team. The Worst-3 inspection team efficiency is 1.9% (3/154) and the Worst-2 inspection team efficiency is 0.6% (1/154). The efficiency improvement for adding a third inspector to Worst-2 inspection team is 1.3% (see top portion of Table 4.27).

When considering the Worst-2 inspection team, the average efficiency improvement for adding a measurer (3.4%) exceeds the efficiency improvement of going from two inspectors to three inspectors, which is of 1.3%. The average net gain is 2.1% (Table 4.27). Fourteen out of 16 measurers have a positive net gain of efficiency over adding a third inspector to the Worst-2 inspection team.

Table 4.27 Efficiency improvement when adding a measurer over the Worst-2 inspection team

Worst-2						
With 3rd inspector J			Inspectors L and E			Increase from 3rd inspector
Defects	Efficiency		Defects	Efficiency		
3	1.9%		1	0.6%		1.3%
Measurer	C&M defects	Dupl.	Added	Effici-ency	Impro-vement	Net gain or loss(-)
C	13	1	12	8.4%	7.8%	6.5%
F	6	1	5	3.9%	3.3%	2.0%
H	6	1	5	3.9%	3.3%	2.0%
J	5	0	5	3.9%	3.3%	2.0%
L	24	0	24	16.2%	15.6%	14.3%
M	3	0	3	2.6%	2.0%	0.7%
N	2	0	2	1.9%	1.3%	0.0%
O	1	0	1	1.3%	0.7%	-0.6%
P	2	0	2	1.9%	1.3%	0.0%
Q	5	0	5	3.9%	3.3%	2.0%
R	3	0	3	2.6%	2.0%	0.7%
T	3	1	2	1.9%	1.3%	0.0%
V	4	0	4	3.2%	2.6%	1.3%
X	3	0	3	2.6%	2.0%	0.7%
Y	1	0	1	1.3%	0.7%	-0.6%
Z	6	0	6	4.5%	3.9%	2.6%
Total	**87**	**4**	**83**	**--**	**--**	**--**
Average	**5.4**	**0.3**	**5.2**	**4.0%**	**3.4%**	**2.1%**
Std dev	**5.5**	**0.4**	**5.5**	**3.6%**	**3.6%**	**3.6%**
Min	**1**	**0**	**1**	**1.3%**	**0.7%**	**-0.6%**
Max	**24**	**1**	**24**	**16.2%**	**15.6%**	**14.3%**

The average number of added defects found by measurers (5.2) compared to the number of defects found by the two inspectors of the Worst-2 team (1), represents a 520.0% increase (5.2/1).

Median-2 team

Adding the next median inspector (inspector H) to the Median-2 inspection team is the same as considering the Median-3 inspection team. The Median-3 inspection team efficiency is 13.0% (20/154) and the Median-2 inspection team efficiency is 9.7% (15/154). The efficiency improvement for adding a third inspector to Median-2 inspection team is 3.3% (see top portion of Table 4.28).

When considering the Median-2 inspection team, the average efficiency improvement for adding a measurer (3.4%) slightly exceeds the efficiency improvement of going from two inspectors to three inspectors, which is of 3.3%. The average net gain is 0.1% (Table 4.28). Five measurers out of 16 have a positive net gain of efficiency over adding a third inspector to the Median-2 inspection team.

Table 4.28 Efficiency improvement when adding a measurer over the Median-2 inspection team

Median-2						
With 3rd inspector H			Inspectors P and C			Increase from 3rd inspector
Defects	Efficiency		Defects	Efficiency		
20	13.0%		15	9.7%		3.3%
Measurer	C&M defects	Dupl.	Added	Effici-ency	Impro-vement	Net gain or loss(-)
C	13	0	13	18.2%	8.5%	5.2%
F	6	0	6	13.6%	3.9%	0.6%
H	6	0	6	13.6%	3.9%	0.6%
J	5	2	3	11.7%	2.0%	-1.3%
L	24	1	23	24.7%	15.0%	11.7%
M	3	0	3	11.7%	2.0%	-1.3%
N	2	0	2	11.0%	1.3%	-2.0%
O	1	0	1	10.4%	0.7%	-2.6%
P	2	0	2	11.0%	1.3%	-2.0%
Q	5	0	5	13.0%	3.3%	0.0%
R	3	0	3	11.7%	2.0%	-1.3%
T	3	0	3	11.7%	2.0%	-1.3%
V	4	0	4	12.3%	2.6%	-0.7%
X	3	0	3	11.7%	2.0%	-1.3%
Y	1	0	1	10.4%	0.7%	-2.6%
Z	6	0	6	13.6%	3.9%	0.6%
Total	**87**	**3**	**84**	**--**	**--**	**--**
Average	**5.4**	**0.2**	**5.3**	**13.1%**	**3.4%**	**0.1%**
Std dev	**5.5**	**0.5**	**5.4**	**3.5%**	**3.5%**	**3.5%**
Min	**1**	**0**	**1**	**10.4%**	**0.7%**	**-2.6%**
Max	**24**	**2**	**23**	**24.7%**	**15.0%**	**11.7%**

The average number of added defects found by measurers (5.3) compared to the number of defects found by the two inspectors of the Median-2 team (15), represents a 35.3% increase (5.3/15).

Best-2 team

Adding the next best inspector (inspector M) to the Best-2 inspection team is the same as considering the Best-3 inspection team. The Best-3 inspection team efficiency is 42.9% (66/154) and the Best-2 inspection team efficiency is 31.2% (48/154). The efficiency improvement for adding a third inspector to Best-2 inspection team is 11.7% (see top portion of Table 4.29).

The average efficiency improvement for adding a measurer to the Best-2 team (3.0%) is below its efficiency improvement of adding a third inspector (11.7%). The net loss is -8.7% (Table 4.29). Only one expert measurer has a positive net gain over the Best-2 inspection team and it is below the efficiency improvement of adding a third inspector.

Table 4.29 Efficiency improvement when adding a measurer over the Best-2 inspection team

Best-2						
With 3rd inspector M			Inspectors O and Q			Increase from 3rd inspector
Defects	Efficiency		Defects	Efficiency		
66	42.9%		48	31.2%		11.7%
Measurer	C&M defects	Dupl.	Added	Effici-ency	Impro-vement	Net gain or loss(-)
C	13	4	9	37.0%	5.8%	-5.9%
F	6	2	4	33.8%	2.6%	-9.1%
H	6	1	5	34.4%	3.2%	-8.5%
J	5	1	4	33.8%	2.6%	-9.1%
L	24	2	22	45.5%	14.3%	2.6%
M	3	0	3	33.1%	1.9%	-9.8%
N	2	0	2	32.5%	1.3%	-10.4%
O	1	0	1	31.8%	0.6%	-11.1%
P	2	0	2	32.5%	1.3%	-10.4%
Q	5	0	5	34.4%	3.2%	-8.5%
R	3	0	3	33.1%	1.9%	-9.8%
T	3	2	1	31.8%	0.6%	-11.1%
V	4	0	4	33.8%	2.6%	-9.1%
X	3	0	3	33.1%	1.9%	-9.8%
Y	1	0	1	31.8%	0.6%	-11.1%
Z	6	0	6	35.1%	3.9%	-7.8%
Total	**87**	**12**	**75**	**--**	**--**	**--**
Average	**5.4**	**0.8**	**4.7**	**34.2%**	**3.0%**	**-8.7%**
Std dev	**5.5**	**1.1**	**4.9**	**3.2%**	**3.2%**	**3.2%**
Min	**1**	**0**	**1**	**31.8%**	**0.6%**	**-11.1%**
Max	**24**	**4**	**22**	**45.5%**	**14.3%**	**2.6%**

The average number of added defects found by measurers (4.7) compared to the number of defects found by the two inspectors of the Best-2 team (48), represents a 9.8% increase (4.7/9.8).

4.5.3.4 Efficiency of measurers without inspection

When inspections are not applied in organizations, every defect found by a measurer systematically increases their efficiency (number of added defects by each measurer compared to the total number of unique defects related to functional requirements found by all participants, which is 154) in finding defects (Table 4.30). Expert measurers (C to L) on average have a greater efficiency (7.0%) than measurers with limited experience (1.9%) as they are able to find a larger number of defects on average (11 defects compared to 3 defects, on average).

Table 4.30 Efficiency of measurers when inspections are not applied

Experience	Measurer	Added defects	Efficiency (added defects/154)	Average efficiency
Experts	C	13	8.4%	7.0% (avg. 11 defects)
	F	6	3.9%	
	H	6	3.9%	
	J	5	3.2%	
	L	24	15.6%	
Limited experience	M	3	1.9%	1.9% (avg. 3 defects)
	N	2	1.3%	
	O	1	0.6%	
	P	2	1.3%	
	Q	5	3.2%	
	R	3	1.9%	
	T	3	1.9%	
	V	4	2.6%	
	X	3	1.9%	
	Y	1	0.6%	
	Z	6	3.9%	
	Average	**5.4**	**3.5%**	
	Std dev	**5.5**	**3.6%**	
	Min	**1**	**0.6%**	
	Max	**24**	**15.6%**	

4.5.4 Unit cost analysis of measurers in finding defects

Measurers expended effort for measuring and identifying defects. The effort of each measurer is added to each inspection team to obtain the total team effort as if the measurer has been added to that team. The unique defects found by each measurer were added to each inspection team. From these new effort and defect values, a new unit cost can be calculated, as well as the variation on each team's unit cost prior to adding a measurer.

To understand the unit cost improvement when adding a measurer as an extra inspection team member, it is important to measure the unit cost of the same teams with an inspector added to them, and next to compare the unit cost variation of each measurer with the unit cost variation of the inspection team including that extra inspector.

The corresponding data analyses are presented in the following subsections.

4.5.4.1 Measurers as a fifth member of an inspection team

When adding the next worst inspector (inspector D) to the Worst-4 team and adding the related effort, the total effort is 501 minutes for 9 defects. The unit cost evolves from 67.2 minutes/defect to 55.7 minutes/defect. The unit cost improvement for adding a fifth inspector to the Worst-4 inspection team is 11.5 minutes/defect (67.2-55.7 minutes/defect) which corresponds to a 17.1% improvement (11.5/67.2 minutes/defect).

When adding the next median inspector (inspector G) to the Median-4 team and adding the related effort, the total effort is 499 minutes for 28 defects. The unit cost evolves from 15.8 minutes/defect to 17.8 minutes/defect. The unit cost does not improve after adding a fifth inspector to the Median-4 inspection team, it deteriorates by -2.0 minutes/defect (15.8-17.8) or -12.7% (-2.0/15.8).

When adding the next best inspector (inspector A) to the Best-4 team and adding the related effort, the total effort is 653 minutes for 88 defects. The unit cost evolves from 7.0 minutes/defect to7.4 minutes/defect. The unit cost does not improve after adding a fifth inspector to the Best-4 inspection team, it deteriorated by -0.4 minutes/defect (7.0-7.4) or -5.7% (-0.4/7.0).

In summary, the unit cost variation between teams of four inspectors and teams of five inspectors is:

- Worst-4 to Worst-5: +11.5 minutes/defect (from 67.2 to 55.7), or +17.1% (11.5/67.2).
- Median-4 to Median-5: -2.0 minutes/defect (from 15.8 to 17.8), or -12.7% (2.0/15.8).
- Best-4 to Best-5: -0.4 minute/defect (from 7.0 to 7.4), or -5.7% (0.4/7.0).

Table 4.31 shows measurers' effort in minutes, the number of defects they add to each team of four inspectors, the new unit cost, and the net gain (or loss) on the unit cost, as if the measurer becomes a fifth inspection team member.

To improve the unit cost of the Worst-4 team, a measurer has to add two or more defects given their average effort. Out of 16 measurers, 11 measurers improve the unit cost of the Worst-4 team with an extra fifth inspector. On average, measurers improve the unit cost of the Worst-4 team by 6.2 minutes/defect, which is better by 9.2% than adding a fifth inspector (6.2/67.2).

On average, measurers decrease the unit cost of the Median-4 team by -0.6 minute/defect (15.8-16.4), which is better by +1.4 minute/defect (17.8-16.4) than adding a fifth inspector or +3.4% (-0.6/17.8).

On average, measurers decrease the unit cost of the Best-4 team with an extra 5th inspector by -0.3 minute/defect (7.4-7.7), which is worst by -4.1% than adding a fifth inspector (-0.3/7.4).

Table 4.31 Variations of unit cost when adding measurers as a fifth inspection team member

		Worst-4			Median-4			Best-4		
With 4 inspectors		**Def-ects**	**Effort**	**Unit cost**	**Def-ects**	**Effort**	**Unit cost**	**Def-ects**	**Effort**	**Unit cost**
		6	403	67.2	26	411	15.8	81	571	7.0
With an extra 5th inspector		**Def-ects**	**Effort**	**Unit cost**	**Def-ects**	**Effort**	**Unit cost**	**Def-ects**	**Effort**	**Unit cost**
		9	501	55.7	28	499	17.8	88	653	7.4
Measurer	Ef-fort	Add-ed	Unit cost	Net gain	Add-ed	Unit cost	Net gain	Add-ed	Unit cost	Net gain
C	127	12	29.4	26.2	12	14.2	3.7	8	7.8	-0.4
F	73	5	43.3	12.4	6	15.1	2.7	4	7.6	-0.2
H	64	5	42.5	13.2	6	14.8	3.0	5	7.4	0.0
J	81	5	44.0	11.7	3	17.0	0.9	4	7.7	-0.3
L	96	24	16.6	39.0	23	10.3	7.5	19	6.7	0.8
N	73	3	52.9	2.8	3	16.7	1.1	2	7.8	-0.3
O	74	2	59.6	-4.0	2	17.3	0.5	1	7.9	-0.4
M	71	1	67.7	-12.0	1	17.9	0.0	1	7.8	-0.4
P	77	2	60.0	-4.3	2	17.4	0.4	2	7.8	-0.4
Q	82	5	44.1	11.6	5	15.9	1.9	5	7.6	-0.2
R	73	3	52.9	2.8	3	16.7	1.1	3	7.7	-0.2
T	75	2	59.8	-4.1	2	17.4	0.5	1	7.9	-0.5
V	87	4	49.0	6.7	3	17.2	0.6	4	7.7	-0.3
X	98	3	55.7	0.0	2	18.2	-0.4	2	8.1	-0.6
Y	112	1	73.6	-17.9	0	20.1	-2.3	1	8.3	-0.9
Z	87	6	40.8	14.8	6	15.6	2.3	6	7.6	-0.1
Average	**84.4**	**5.2**	**49.5**	**6.2**	**4.9**	**16.4**	**1.5**	**4.3**	**7.7**	**-0.3**
Std dev	**16.1**	**5.5**	**13.7**	**13.7**	**5.4**	**2.1**	**2.1**	**4.3**	**0.3**	**0.3**
Min	**64**	**1**	**16.6**	**-17.9**	**0**	**10.3**	**-2.3**	**1**	**6.7**	**-0.9**
Max	**127**	**24**	**73.6**	**39.0**	**23**	**20.1**	**7.5**	**19**	**8.3**	**0.8**

4.5.4.2 Measurers as a fourth member of an inspection team

Table 4.32 shows measurers' effort, the number of defects they add to each team of three inspectors, the new unit cost, and the net gain (or loss) on unit cost, as if the measurer became a fourth inspection team member.

The unit cost variation between teams of three inspectors and teams of four inspectors is:

- Worst-3 to Worst-4: +30.2 minutes/defect (97.3 to 67.2), or +31.0% (+30.2/97.3).
- Median-3 to Median-4: +0.8 minute/defect (16.6 to 15.8), or +4.8% (+0.8/16.6).
- Best-3 to Best-4: -0.2 minute/defect (6.8 to 7.0), or -2.9% (-0.2/6.8).

To improve the unit cost of the Worst-3 team, a measurer has to add one or more defects given their average effort. Out of 16 measurers, 11 measurers improve the unit cost of the Worst-3 team with an extra 4th inspector. The average unit cost improvement of adding one measurer to the Worst-3 team was of +9.6 minutes/defect, which exceeds adding a fourth inspector by +9.9% (9.6/97.3).

Four measurers out of 16 improve the unit cost of Median-3 team, but the unit cost of the Meadian-3 team deteriorates by -1.3 minute/defect on average, which is a variation of -7.8% compared to adding a fourth inspector (-1.3/16.6).

Except for measurer L who has found a much larger number of new defects than the other measurers, all measurers deteriorate the Best-3 team's unit cost. The average unit cost increase is -0.5 minute/defect, which is a variation of -7.4% compared to adding a fourth inspector (-0.5/6.8).

Table 4.32 Variations of unit cost when adding measurers as a fourth inspection team member

		Worst-3			Median-3			Best-3		
With 3 inspectors		**Def-ects**	**Effort**	**Unit cost**	**Def-ects**	**Effort**	**Unit cost**	**Def-ects**	**Effort**	**Unit cost**
		3	292	97.3	20	332	16.6	66	446	6.8
With an extra 4th inspector		**Def-ects**	**Effort**	**Unit cost**	**Def-ects**	**Effort**	**Unit cost**	**Def-ects**	**Effort**	**Unit cost**
		6	403	67.2	26	411	15.8	81	571	7.0
Measurer	Ef-fort	Add-ed	Unit cost	Net gain	Add-ed	Unit cost	Net gain	Add-ed	Unit cost	Net gain
C	127	12	27.9	39.2	13	13.9	1.9	9	7.6	-0.6
F	73	5	45.6	21.5	6	15.6	0.2	4	7.4	-0.4
H	64	5	44.5	22.7	6	15.2	0.6	5	7.2	-0.1
J	81	5	46.6	20.5	3	18.0	-2.1	4	7.5	-0.5
L	96	24	14.4	52.8	23	10.0	5.9	21	6.2	0.8
N	73	3	60.8	6.3	3	17.6	-1.8	3	7.5	-0.5
O	74	2	73.2	-6.0	2	18.5	-2.6	2	7.6	-0.6
M	71	1	90.8	-23.6	1	19.2	-3.4	1	7.7	-0.7
P	77	2	73.8	-6.6	2	18.6	-2.8	2	7.7	-0.6
Q	82	4	53.4	13.7	4	17.3	-1.4	4	7.5	-0.5
R	73	3	60.8	6.3	3	17.6	-1.8	3	7.5	-0.5
T	75	2	73.4	-6.2	3	17.7	-1.9	1	7.8	-0.7
V	87	4	54.1	13.0	3	18.2	-2.4	4	7.6	-0.6
X	98	3	65.0	2.2	2	19.5	-3.7	3	7.9	-0.8
Y	112	1	101.0	-33.8	0	22.2	-6.4	1	8.3	-1.3
Z	87	6	42.1	25.1	6	16.1	-0.3	6	7.4	-0.4
Average	**84.4**	**5.1**	**58.0**	**9.2**	**5.0**	**17.2**	**-1.4**	**4.6**	**7.5**	**-0.5**
Std dev	**16.1**	**5.5**	**21.5**	**21.5**	**5.5**	**2.6**	**2.6**	**4.7**	**0.4**	**0.4**
Min	**64**	**1**	**14.4**	**-33.8**	**0**	**10.0**	**-6.4**	**1**	**6.2**	**-1.3**
Max	**127**	**24**	**101.0**	**52.8**	**23**	**22.2**	**5.9**	**21**	**8.3**	**0.8**

4.5.4.3 Measurers as a third member of an inspection team

Table 4.33 shows measurers effort, the number of defects they add to each team of two inspectors, the new unit cost, and the variation of unit cost, as if the measurer became a third inspection team member.

The unit cost variation between teams of two inspectors and teams of three inspectors was:

- Worst-2 to Worst-3: +115.7 minutes/defect (213.0 to 97.3), or +54.3% (115.7/213.0).
- Median-3 to Median-4: -0.1 minute/defect (16.5 to 16.6), or -0.4% (-0.1/16.5).
- Best-3 to Best-4: -0.4 minute/defect (6.4 to 6.8), or -6.3% (-0.4/6.4).

All 16 measurers improve the unit cost of the Worst-2 team, mainly due to the fact that the Worst-2 team has found only one defect. The average unit cost was 71.7 minutes/defect for an improvement of 141.3 minutes/defect (213.0-71.7), which exceeds adding a third inspector by 25.6 minutes/defect (97.3-71.7).

Five measurers out of 16 improve the unit cost of Median-2 team. The average unit cost of measurers with the Median-2 team is 17.1 minutes/defect, deteriorates by -0.6 minute/defect (16.5-17.1), which is worst than adding a third inspector by -0.5 minute/defect (16.6-17.1).

Except for measurer L, all other measurers deteriorate the Best-2 team's unit cost. The average deterioration of unit cost for the Best-2 team was of -0.7 minute/defect, which is a variation of -10.9% compared to adding a third inspector (-0.7/6.4). However, the average unit cost of 7.4 minutes/defect with measurers only has 1.0 minute/defect of difference, which may not be of significant value in a decision for adding or not a measurer in an inspection team.

Table 4.33 Variations of unit cost when adding measurers as a third inspection team member

		Worst-2			Median-2			Best-2		
With 2 inspectors		**Def-ects**	**Effort**	**Unit cost**	**Def-ects**	**Effort**	**Unit cost**	**Def-ects**	**Effort**	**Unit cost**
		1	213	213.0	15	248	16.5	48	305	6.4
With an extra 3rd inspector		**Def-ects**	**Effort**	**Unit cost**	**Def-ects**	**Effort**	**Unit cost**	**Def-ects**	**Effort**	**Unit cost**
		3	292	97.3	20	332	16.6	66	446	6.8
Measurer	Ef-fort	Add-ed	Unit cost	Net gain	Add-ed	Unit cost	Net gain	Add-ed	Unit cost	Net gain
C	127	12	26.2	71.2	13	13.4	3.2	9	7.6	-0.8
F	73	5	47.7	49.7	6	15.3	1.3	4	7.3	-0.5
H	64	5	46.2	51.2	6	14.9	1.7	5	7.0	-0.2
J	81	5	49.0	48.3	3	18.3	-1.7	4	7.4	-0.7
L	96	24	12.4	85.0	23	9.1	7.5	22	5.7	1.0
N	73	3	71.5	25.8	3	17.8	-1.2	3	7.4	-0.7
O	74	2	95.7	1.7	2	18.9	-2.3	2	7.6	-0.8
M	71	1	142.0	-44.7	1	19.9	-3.3	1	7.7	-0.9
P	77	2	96.7	0.7	2	19.1	-2.5	2	7.6	-0.9
Q	82	4	59.0	38.3	4	17.4	-0.8	4	7.4	-0.7
R	73	3	71.5	25.8	3	17.8	-1.2	3	7.4	-0.7
T	75	2	96.0	1.3	3	17.9	-1.3	1	7.8	-1.0
V	87	4	60.0	37.3	4	17.6	-1.0	4	7.5	-0.8
X	98	3	77.8	19.6	3	19.2	-2.6	3	7.9	-1.1
Y	112	1	162.5	-65.2	1	22.5	-5.9	1	8.5	-1.8
Z	87	6	42.9	54.5	6	16.0	0.6	6	7.3	-0.5
Average	**84.4**	**5.2**	**71.7**	**25.6**	**5.3**	**17.1**	**-0.5**	**4.7**	**7.4**	**-0.7**
Std dev	**16.1**	**5.5**	**38.6**	**38.6**	**5.4**	**3.0**	**3.0**	**4.9**	**0.6**	**0.6**
Min	**64**	**1**	**12.4**	**-65.2**	**1**	**9.1**	**-5.9**	**1**	**5.7**	**-1.8**
Max	**127**	**24**	**162.5**	**85.0**	**23**	**22.5**	**7.5**	**22**	**8.5**	**1.0**

4.5.4.4 Unit cost of measurers without inspection

When inspections are not applied within an organization, the average unit cost of measurers is 29.4 minutes/defect (Table 4.34). Expert measurers (C to L) tend to have a better unit cost (10.6) than measurers with limited experience (38.0) as they are able to find a larger number of defects, on average.

Table 4.34 Unit cost of measurers without inspection

Experience	Measurer	Effort (minutes)	Defects	Unit cost (effort/defects)	Average unit cost (min./defect)
Experts	C	127	13	9.8	10.6 (avg. 11defects)
	F	73	6	12.2	
	H	64	6	10.7	
	J	81	5	16.2	
	L	96	24	4.0	
Limited	M	73	3	24.3	38.0 (avg. 3 defects)
	N	74	2	37.0	
	O	71	1	71.0	
	P	77	2	38.5	
	Q	82	5	16.4	
	R	73	3	24.3	
	T	75	3	25.0	
	V	87	4	21.8	
	X	98	3	32.7	
	Y	112	1	112.0	
	Z	87	6	14.5	
	Average	**84.4**	**5.4**	**29.4**	
	Std dev	**16.1**	**5.5**	**26.3**	
	Min	**64**	**1**	**4.0**	
	Max	**127**	**24**	**112.0**	

4.5.5 Summary of efficiency and unit cost analyses

4.5.5.1 Summary of efficiency analyses

On average, adding a measurer to an inspection team increases the efficiency of the team. But when a measurer replaces an extra inspector who would have been added to the inspection team, results show an average net loss of -1.2%. The standard deviation being higher than the average suggested that the sole net gain or loss value might not be sufficient to decide whether or not adding a measurer to existing inspection teams when inspecting functional requirements. Another factor to consider would be the value of FSM results provided by a measurer as a management input for decision making within an organization.

The efficiency analysis is summarized in Table 4.35. All measurers provided the functional size of the software and they all added new defects that were overlooked by inspectors.

Table 4.35 Summary of efficiency analysis

Efficiency					
Inspection team	**Effi-ciency**	**Average with one measurer**	**Impro-vement**	**with extra inspector**	**Net gain or loss (-)**
Worst-2	0.6%	4.0%	3.4%	1.9%	2.1%
Worst-3	1.9%	5.3%	3.4%	3.9%	1.4%
Worst-4	3.9%	7.2%	3.3%	5.8%	1.4%
Median-2	9.7%	13.1%	3.4%	13.0%	0.1%
Median-3	13.0%	16.3%	3.3%	16.9%	-0.6%
Median-4	16.9%	20.1%	3.2%	18.2%	1.9%
Best-2	31.2%	34.2%	3.0%	42.9%	-8.7%
Best-3	42.9%	45.9%	3.0%	52.6%	-6.7%
Best-4	52.6%	55.3%	2.7%	57.1%	-1.8%
Average	**19.2%**	**22.4%**	**3.2%**	**23.6%**	**-1.2%**
Std dev.	**17.7%**	**17.5%**	**0.2%**	**20.3%**	**3.7%**
Measurers only	0%	3.5%	3.5%	0%	3.5%

Results of measurers without inspectors show an average efficiency of 3.5% per measurer, compared to 0.0% efficiency in the absence of inspections. However, this result is clearly below the efficiency of inspection teams – except for Worst-2 and Worst-3 teams – when several inspectors participate in finding defects.

4.5.5.2 Summary of unit cost analyses

Table 4.36 shows that, on average, adding a measurer to an inspection team improves their unit cost by 9.5%. However, there is a higher net gain over worst teams, composed by those inspectors having found the lowest number of defects. When considering inspection teams composed of experts (Best teams), there is a higher relative net gain of adding an extra expert inspector than an average measurer. But the absolute net loss of adding a measurer over a team of expert inspectors is below 1.3 minute/defect, which might not be significant for an organization interested in obtaining the functional size from the software requirements.

Table 4.36 Summary of unit cost analyses

Unit cost							
Inspection team	**Unit cost**	**with one measurer**	**Improvement**		**with extra inspector**	**Net gain or loss(-) of measurers**	
Worst-2	213.0	71.7	141.3	66.3%	97.3	25.6	26.4%
Worst-3	97.3	57.6	39.8	40.9%	67.2	9.6	14.3%
Worst-4	67.2	49.5	17.7	26.3%	55.7	6.2	11.1%
Median-2	16.5	17.1	-0.6	-3.7%	16.6	-0.5	-3.3%
Median-3	16.6	17.2	-0.6	-3.3%	15.8	-1.3	-8.5%
Median-4	15.8	16.4	-0.5	-3.5%	17.8	1.5	8.2%
Best-2	6.4	7.4	-1.1	-17.0%	6.8	-0.7	-10.0%
Best-3	6.8	7.5	-0.8	-11.5%	7.0	-0.5	-6.9%
Best-4	7.0	7.7	-0.7	-9.3%	7.4	-0.3	-3.8%
Average	**49.6**	**28.0**	**21.6**	**9.5%**	**32.4**	**4.4**	**3.1%**
Std dev.	**65.1**	**23.3**	**44.3**	**26.9%**	**31.0**	**8.3**	**11.8%**
Measurers only	29.4	--	-7.8	-26.5%	--	--	--

Results of measurers without inspectors show an average unit cost of 29.4 minutes/defect. This result is between the unit cost of worst and median teams of inspectors. From the unit cost point of view, there is an advantage of including a measurer within an inspection team rather than letting measurers alone for identifying defects while measuring the software size. The average unit cost improvement of adding a measurer to an inspection team is 21.6 minutes/defect, which is better by 7.8 minutes/defect (21.6-29.4) over the measurers only, or 26.5% improvement (7.8/29.4).

All measurers provide a functional size with a list of defects and issues within the same time-frame than the inspectors take to only identify defects.

4.6 Defect analysis

4.6.1 Nature of defects found by inspectors

Other than spelling and syntax defects, inspectors found a variety of critical and minor defects, all of which related to at least one of the defined rules that was violated. The most common critical defects are related to:

1) Inconsistencies, such as use case descriptions not matching those of the use case model. Defects of this nature potentially affect the quality of the resulting software and documentation as one may refer to a certain naming while the other uses a different naming.
2) Missing information, such as completely missing use case descriptions that were visible in the use case model, or links between use cases that should have been visible in the use case model. Defects of this nature may be interpreted incorrectly by developers, leading to undesired software behaviour.
3) Ambiguities, such as the security handling, or the description of event types that were being sent by the client application to the server application. Defects of this nature may result in developers taking inappropriate decisions, leading to undesirable software behaviour.

4) Incorrect information, such as the wrong order of steps within use cases. Defects of this nature may affect data integrity and software behaviour.

The most common minor defects are related to:

1) Inconsistencies, such as the usage of the passive voice in some functional requirements when most of them used the active voice.
2) Missing information, such as the numbering of steps within some use cases (these defects were a result of a malfunction from the word processor printing function, as the numbering is present in the electronic file but does not appear on printed paper), or brief description section of a use case missing while the use case title is self-explanatory.
3) Ambiguities, such as inadequate choice of words to describe user interface elements.
4) Incorrect information, such as design details about buttons that should have been kept at a higher level.

Minor defects are, by nature, easy to fix and the impact of these defects must be small. Otherwise, they would have been typed as critical, not minor.

4.6.2 Nature of defects found by measurers

There is one element that distinguishes measurers from inspectors: measurers are knowledgeable about the COSMIC method. As they apply this FSM method, their measurement task over the SRS is accomplished from a perspective completely different from the perspective of the inspectors. This different perspective is reflected in the nature of defects found by the measurers, although the same quality attributes were applied to the SRS document. In addition to a total of 29 unique functional defects (C, M, and S) found by both inspectors and measurers, 53 unique functional defects were found only by measurers. Among these unique functional defects, the most common critical defects were related to:

1) Inconsistencies, such as a wrong functional decomposition where several use cases decomposition did not match the definition of a functional process from the COSMIC method or the UML criteria for use case decomposition. Moreover, defects of this nature

have an impact on the number of use cases, the functional size of the software, and, more importantly, the overall understanding of the software functionalities. As an example, the 'Disconnect' feature was defined as an extension point of the 'Connect' use case, when it should have been defined in its own use case and separately in the two boundaries (one use case on the client and one use case on the server).

2) Missing information, such as the absence of a data model. A defect of this nature was leading inspectors and measurers to identify other defects related to ambiguities about data objects being manipulated by the software. Another example is missing use cases related to client-server clocks synchronization. No inspector saw this defect but a measurer did as he followed the path of timing related data to ensure integrity of FSM results. This was a defect affecting FSM results and software behaviour.
3) Ambiguities, such as the number of files that were read from and written to, as each one was a potential data group (e.g. one file for audio and one file for video, or one single file for audio/video). Therefore, defects of this nature had a direct impact on FSM results and they may be misleading developers.

4.6.3 Level of details of identified defects

Inspectors and measurers reported critical and minor defects at different levels of detail. From a statistical viewpoint, all defects have the same weight. However, the impact of each defect on the software project may vary greatly with respect to cost or quality in use (i.e. fit for purpose). For that reason, quantitative analysis of defects might be insufficient and qualitative analysis appeared to be appropriate, such as identifying the level of detail of each defect. Some were at a 'high-level' as they refer to the whole SRS document or an entire use case or an entire scenario or several use case steps. Others were at a 'low-level' as they were very specific about the line and the words within a line that was causing the defect. This classification was applied to each defect, regardless of their type, based on these definitions.

For high-level defects, it was possible to relate several identified low-level defects, such as:

1) No data models [defect raised at the SRS level].
   a) The number of files to close not specified when actor issues a closing command [in use case ‘Close uSleuth’].
   b) Configuration of parameters file: What's in it? [in use case ‘Connect uSpy to uSleuth’].
   c) The event log file… Unclear [in use case ‘Record an experiment’].
2) Missing communication links between use cases.
   a) Indication for a link between ‘Playback’ and ‘Open’? [in figure 5].
3) Not clear [being identified for the alternate flow ‘Pause’ of the ‘Playback’ use case].
   a) Where is this step in the main flow? Returns to ‘Alternate flow-Pause, step 5’ where it came from?? [at lines 8 to 10 of the alternate flow ‘Pause’].
   b) Alternate command part of triggering event? [at line 2 of the alternate flow ‘Pause’].
   c) Event and image may not match [at line 6 of the alternate flow ‘Pause’].
4) Use cases missing.
   a) Disconnect: described in a separate UC? Same function, different content of the data groups [at extension point of the ‘Connect use case’].
   b) This is not in the ‘Record’ state, recording has stopped. Different functional process? Trigger for another UC or extension? [at lines 11 to 13 of the ‘Record’ use case].
   c) 13 UC in Fig.3, but only 10 UC descriptions [at figure 3].

From these observations, each of the 356 confirmed defects have been classified as either ‘high-level’ or ‘low-level’. Of these, 9.8% (35/356) were high-level and 90.2% (321/356) were low-level defects. Out of the 321 low-level defects, 93 (29%) were found by measurers only, 190 (59%) were found by inspectors only, and 38 (12%) were found by both. Out of the 35 high-level defects, 21 (60%) were found by measurers only, 9 (26%) were found by inspectors only, and 5 (14%) were found by both. Thus, the number of high-level defects found by measurers only is 50% higher than the number of high-level defects found by inspectors (21/(9+5)=21/14→50% higher), as each high-level defect may lead to identify several low-level defects. No further analysis was done on these defects.

4.6.4 Identification of new defects through experimental sessions

As the experimental sessions took place, new unique confirmed defects were identified by participants. Once all defects were collected from the 35 participants, analysis was done to understand how many of these new defects were discovered by participants in each experimental session. Figure 4.4 shows the distribution of the 244 uniquely identified defects and issues in the whole SRS (excluding the rejected defects and issues). These numbers corroborate those of Table 4.7 (170 defects and 74 issues). Figure 4.5 shows the distribution of the 109 uniquely identified critical and minor defects related to FR only. These numbers corroborate those of Table 4.10 (45 critical and 64 minor defects).

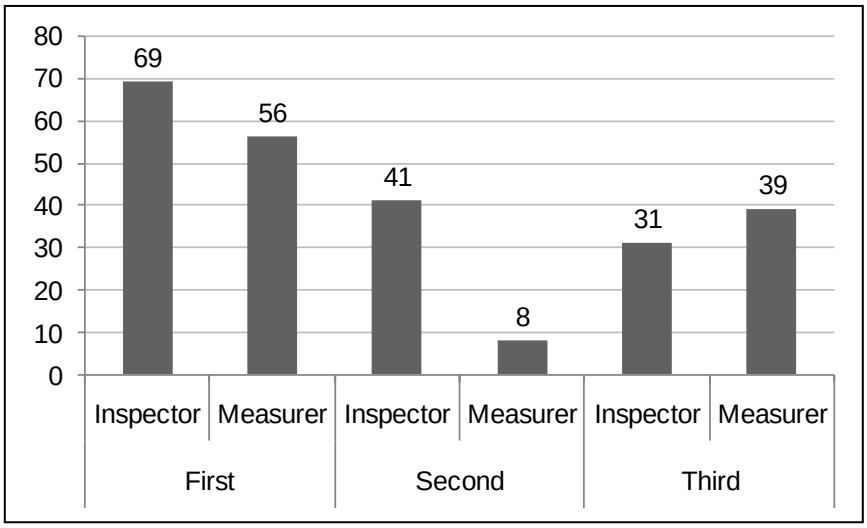

Figure 4.4 New uniquely identified defects in each experimental session

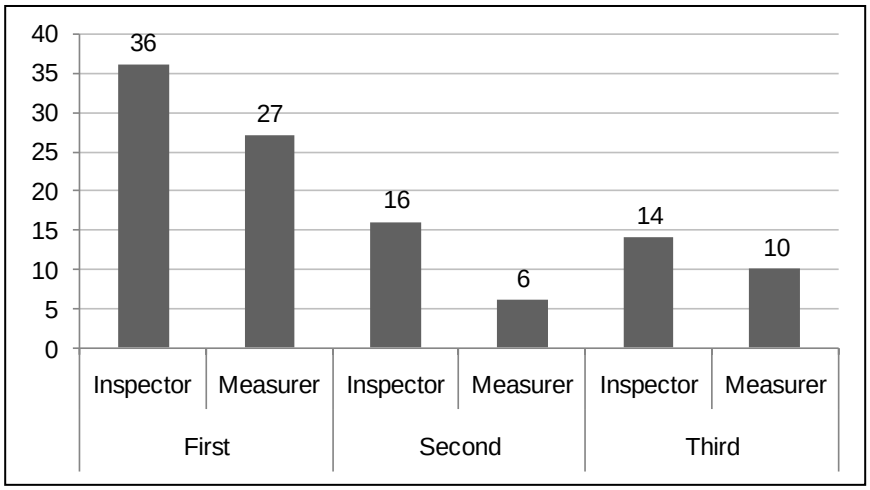

Figure 4.5 New uniquely identified critical and minor defects related to FR only, in each experimental session

The majority of experts participated in the first experimental session, which explains the higher values of the related columns. The second experimental session was composed of a majority of participants with limited experience, having only one expert inspector among the participants, which explains the lower values for the second experimental session.

A significant proportion of unique defects were identified by participants in every experimental session. These results suggest that only one inspection cycle would be insufficient and that several inspection cycles would be required in order to eliminate the majority of defects. Several inspections cycles might not be economically viable as it would delay the software product release.

Thus, how many inspections would it require for an SRS document to be extensively verified in order to eliminate all critical and minor defects impacting the measured functional size? This research does not answer that question directly as many successive inspections would have been required through many experiments to answer that question. However, the data suggest strong evidence of residual defects after a single inspection has been completed.

Defects related to functional requirements have a potential impact on the functional size and it would be of a good practice for a measurer to identify these defects while measuring, and

analyze what would be their impact on the functional size. This issue is addressed in the next chapter.

4.6.5 Analysis of defects potentially affecting FSM results

This section briefly describes the identified critical and minor defects that had or may have had an impact on the resulting functional size. Each defect in functional requirements may lead to a size difference of several CFP (COSMIC Function Points).

1) Unclear functional description

Among the 73 functional critical defects, 26 were related to unclear functional behaviour description, stating that more details would be required. These ambiguities may have resulted in different interpretation of the functional requirements, and thus a different functional size.

2) Missing functional processes

One specific use case, identified as a functional process, had an extension point that should have been defined as a separate use case (‘Disconnect uSpy from uSleuth’), thus a separate functional process. Five measurers did not identify this functional process, which contributed to an imprecise functional size.

3) Missing error handling

It is common that error handling in a functional process requires additional DMs, specifically when comparing a value inside the functional process with a value outside the functional process, either from persistent storage – a Read (R) DM – or from an engineered device or external system – an eXit (X) DM for the request and an Entry (E) DM for the answer. Only one measurer raised a critical defect for a specific missing error handling in a functional process, but four measurers applied FSM as if the error handling was described; it was not. Therefore, size measurement for this particular functional process showed two to three

irrelevant DMs in these four cases, since it was clearly stated in the document that no error handling would be done in this software intended for a proof of concept. Again, this ambiguity within a functional process description led the measurers to an imprecise functional size.

4) Ambiguities of data groups

The SRS was unclear about the manipulation of a specific data group: the audio/video file could have been interpreted as two different data groups. Since this data group required several movements (Read, Write, and eXit) in seven functional processes, doubling it had resulted in superfluous DMs.

5) Ambiguity related to multiple occurrences

One functional process was sending multiple occurrences of the same data group to a software outside its boundary. From a developer's point of view, the related requirements were clear, but from one measurer's point of view, this was perceived as a continuous flow, which it was not. This measurer raised a minor defect related to a measurement ambiguity since he felt unable to measure the size of the perceived continuous flow of this data group. As a consequence, he did not measure that functional process at all, which resulted in an incomplete functional size.

CHAPTER 5

PHASES 3 & 4: THE INFLUENCE OF DEFECTS ON FSM RESULTS

5.1 Functional size data from Phase 2 experiments

During the three experimental sessions of Phase 2, measurers applied the COSMIC method on uObserve SRS v1.0 and provided their FSM results. The measurement data are presented separately for expert participants and for participants with limited experience in order to do a comparative analysis.

5.1.1 FSM data from experts

Functional size measurement results in COSMIC Function Point (CFP) show variations among expert measurers (Table 5.1).

Table 5.1 FSM results per expert measurer

Measurer	Number of functional processes	Functional size (in CFP)
C	14	57
F	10	62
H	11	55
J	10	55
L	14	61
Average	**11.8**	**58**
Standard deviation	**1.8**	**3**

Some of these variations in the sizes might be due to defects in the SRS document: indeed, there were defects identified related to the functional decomposition of the software. For instance, the SRS contains 10 use cases for which a detailed description is available. Two measurers identified these as the only 10 functional processes to measure. Three measurers identified an 11th functional process (‘Disconnect uSpy from uSleuth’) that was described as an extension point of a use case (‘Connect uSpy to uSleuth’). According to the COSMIC

method definition of a functional process, it was indeed a separate functional process that was missed by measurers F and J due to a defect in the functional decomposition in the SRS document.

Measurer C also identified three additional functional processes from the written functional requirements for ‘Stop recording’, ‘Stop playing back’, and ‘Pause playing back’, for a total of 14 functional processes: the reason was that the software human user was pressing on a ‘Stop’ or ‘Pause’ button which he considered as triggering events from outside the software boundary. This was his interpretation of the COSMIC method definitions of a ‘functional process’ and a ‘triggering event’. This specific case may have ambiguities related to the last sentence of the functional process definition saying ‘it is complete when it has executed all that is required to be done in response to the triggering event’. The ‘Stop’ and ‘Pause’ buttons could be viewed as elements by which the user triggers independent functional process but it is unclear in this case if the functional process that was being executed (i.e. ‘Record an experiment’ and ‘Playback an experiment’) had executed ‘all that was required to be done in response to its triggering event’. Maybe not for the ‘Record an experiment’ use case as it required to be stopped at one point in time to record and close the data files adequately in order to do all that was required in response to its triggering event (i.e. the user wanted to record an experiment by pressing the ‘Record button’).

Another measurer identified 14 functional processes that were part of a specific use case but should have been described in separate use cases, thus being separate functional processes. Again, this derived from defects in the functional decomposition of the SRS document.

Measurer L provided a possible interval of the measurement results based on potential defect corrections, where the final size could vary between 52 and 62 CFP. However, the recorded size of 61 CFP was verified by himself and a colleague and was compliant with his measurement notes from his experimental session.

5.1.2 FSM data from measurers with limited experience

FSM results from measurers with limited experience contain larger variations in the functional size than FSM results from experts (Table 5.2). Only three limited experience measurers obtained results within 20% of experts' average (T, W, and Y). Measurers R and S are excluded from the calculation of average and standard deviation because they have not followed the experimental procedure.

Table 5.2 FSM results measurers with limited experience

Measurer	Number of functional processes	Functional size (in CFP)	Difference with experts' average (in CFP)	% of difference (from 58.0 CFP)
M	11	73	+15	25.9%
N	11	73	+15	25.9%
O	11	77	+19	32.8%
P	10	73	+15	25.9%
T	10	56	-2	-3.4%
V	10	45	-13	-22.4%
W	12	61	+3	5.2%
X	9	37	-21	-36.2%
Y	8	48	-10	-17.2%
Z	9	43	-15	-25.9%
Q	12	136	+78	134.5%
R	11	189	+131	225.9%
S	4	30	-28	-48.3%
Average:	**10.1**	**58.6**	**+0.6**	**1.0%**
Std dev.:	**1.1**	**14.1**	**14.1**	**24.3%**
Min:	**8**	**37**	**-21**	**-36.2%**
Max:	**12**	**77**	**+19**	**32.8%**

Measurers Q and R did not apply the COSMIC method adequately by measuring DMs on multiple occurrences of data group manipulations, and by measuring movements on navigation actions that did not convey data. Hence, their data are excluded from calculation of average and standard deviation. Measurer S measured only four out of the 10 defined use cases and his measurement was mostly wrong when compared to experts' measurement

results. Because of incomplete measurement, his data were excluded from the average and standard deviation and are also excluded from data analysis.

On average, there are smaller variations in the number of functional processes from limited experience measurers than from expert measurers. The reason may be that less experienced measurers tend to identify a one-to-one relationship between a detailed use case and a functional process, assuming that the functional decomposition in the SRS document is accurate.

5.1.3 FSM analyses

5.1.3.1 Analysis of FSM result differences among expert measurers

Measurer J having identified 10 functional processes recorded that if functional processes are not written in the requirements document being used as a basis for FSM activity, then they should not be measured. Two other experts seem to disagree with that statement as they identified defects related to the functional decomposition that led them to identify functional processes that were not described at all or not clearly described as being separate functional processes. The COSMIC method provides guidance on that matter in the 'Measurement Strategy' phase, where the measurement purpose is defined and used as a guiding principle for the measurement activity. The experimental procedure used in the first, second and third experimental sessions provided guidance for defect identification but was not explicit on how to apply the measurement on potential solutions to these defects.

5.1.3.2 FSM quality challenges for measurers with limited experience

Significant differences were observed among measurement results of all measurers having limited experience. The reported functional size varied from 37 CFP to 77 CFP when excluding measurers Q, R, and S (see Table 5.2). The correct size interval, as measured by experts, was ranging from 54 CFP to 62 CFP in the presence of a large number of defects in

the SRS document. Variations in the sizes obtained by measurers with limited experience might be due to misunderstanding of the COSMIC method and to defects in the SRS document. This section identifies inconsistencies of their FSM results against the COSMIC method that had an impact on the resulting functional size.

1) Incorrect identification of functional processes and related Entry DMs

The SRS document contained between 10 and 14 functional processes (according to experts), but not all measurers identified them correctly. In the uObserve SRS v1.0, each use case corresponded to at least one functional process and a triggering event was identified for each one. Triggering events are normally associated with an Entry movement. A significant number of triggering Entry movements for identified functional processes was missing (Table 5.3).

Table 5.3 Missing entry movements of triggering events

Measurer	# FP identified	# Triggering Entry	# Missing Entry	% missing from total
M	11	11	0	0%
N	11	11	0	0%
O	11	11	0	0%
Q	12	11	1	8%
P	10	3	7	70%
R	11	1	10	91%
S	4	4	0	0%
T	10	3	7	70%
V	10	1	9	90%
W	12	12	0	0%
X	9	9	1	10%
Y	8	5	3	38%
Z	9	3	6	67%
Average:	**9.8**	**6.5**	**3.4**	**34%**
Std dev.:	**2.0**	**4.1**	**3.7**	**36%**

These missing Entry movements represent an equal number of missing CFP in the total size reported by participants. Results are showing extremes: participants either missed none or a

few, or they missed a lot (over 2/3). Experts have not missed triggering entry at all. This challenge with measurers of limited experience could be addressed with an element in a measurement checklist.

2) Incorrect identification of data groups

A challenging task of FSM is to correctly identify data groups that are manipulated by each functional process. However, this task is much easier when requirements are unambiguous, including consistency in data groups naming.

For nine functional processes, participants identified several data groups where only one should have been found. As an example, one data group per data attribute (of the same data group) was identified in several cases. In other cases, measurers referred to data groups not mentioned in the requirements and related DMs were wrongly counted. In all these cases, the resulting size included superfluous DMs that should be removed, which would reduce the total size reported by concerned participants.

Many participants identified navigational elements (e.g. buttons or cursors) as data groups when these elements were not carrying any data. They measured Entry and eXit DMs related to these elements that were superfluous.

3) Missing data groups

Putting aside missing functional processes, several existing data groups were not identified in several functional processes. DMs associated to these missing data groups were also missing, which resulted in smaller functional sizes.

4) Wrong DMs

In numerous cases where data groups were consistently identified with the requirements text, corresponding DMs were not. As an example, one measurer systematically identified a Write movement where it clearly should have been an eXit as the data carried by these data groups were displayed on the user interface. Nevertheless, misclassification has little impact on the resulting size since the COSMIC method assigns the same size to all four data movement types.

5) Missing DMs

DMs have been missed on 13 of the 15 identified functional processes by all measurers with limited experience. Any missing DM should have been added to the reported functional size.

6) Duplicate or superfluous DMs

All measurers with limited experience identified superfluous DMs within 11 of identified functional processes. These include 13% of identified functional processes showing a DM type repeated at least once for the same data group, where the DM type should had been identified and considered only once per data group within a functional process.

There were also many superfluous DMs that could not be traced back to the requirements text, i.e. the data group was mentioned, the adequate data groups were identified but participants also identified related DMs that do not exist at all in the requirement text without being a defect.

All superfluous DMs should have been subtracted from the reported functional size.

5.2 Overview of Phase 3 and Phase 4 experiments

Phases 3 and 4 of the research project addressed the fourth research objective: determine the influence of defects on functional size. Phase 3 was setup to prepare experimental material to use during the Phase 4 experimental sessions. Two experimental sessions were planned and held in Phase 4 in order to gather data related to the forth research objectives (Figure 5.1). A total of 10 measurers participated in these experiments.

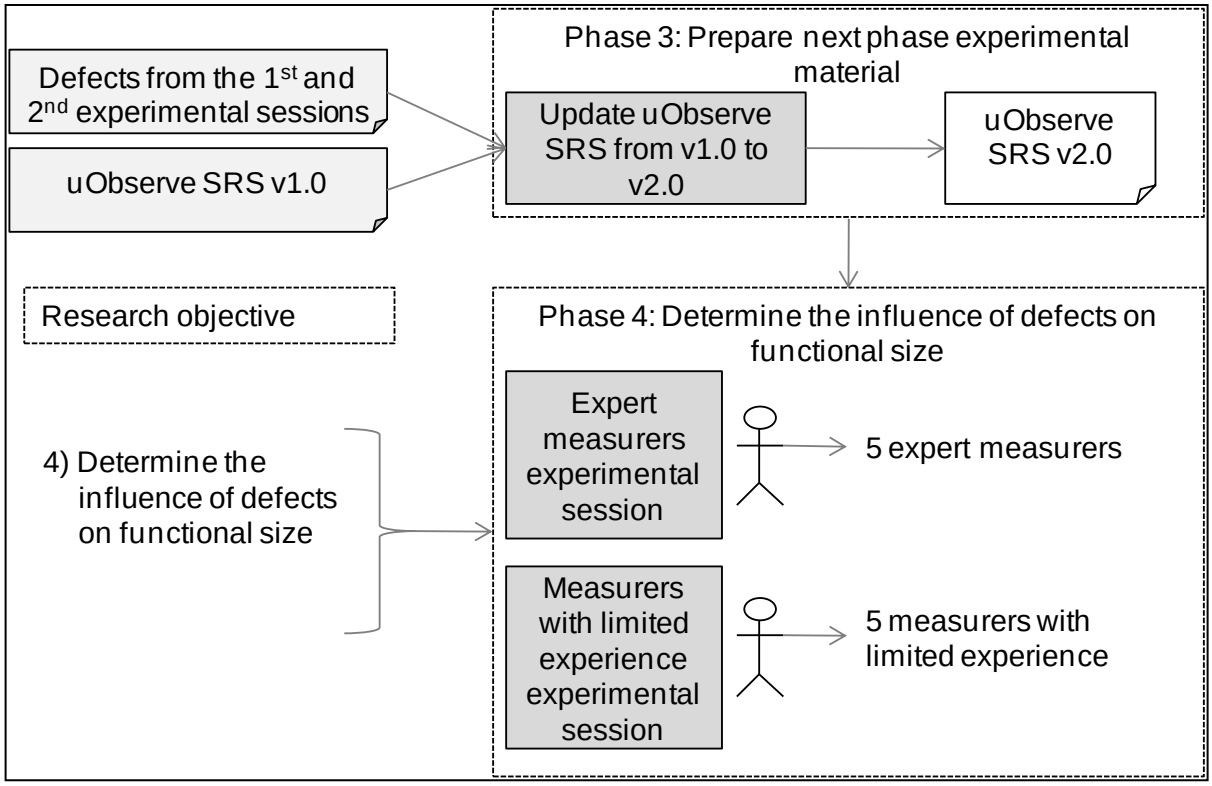

Figure 5.1 Overview of Phase 3 and Phase 4 experiments

The expert measurer experimental session was held during summer of 2011 and targeted the same expert measurers who participated in the first and third experimental sessions. The measurer with limited experience experimental session was held in the same symposium workshop as the third experimental session from Phase 2, in August 2010. Therefore, Phase 3 happened while Phase 2 was still active, after the first and second experimental sessions (Figure 5.2).

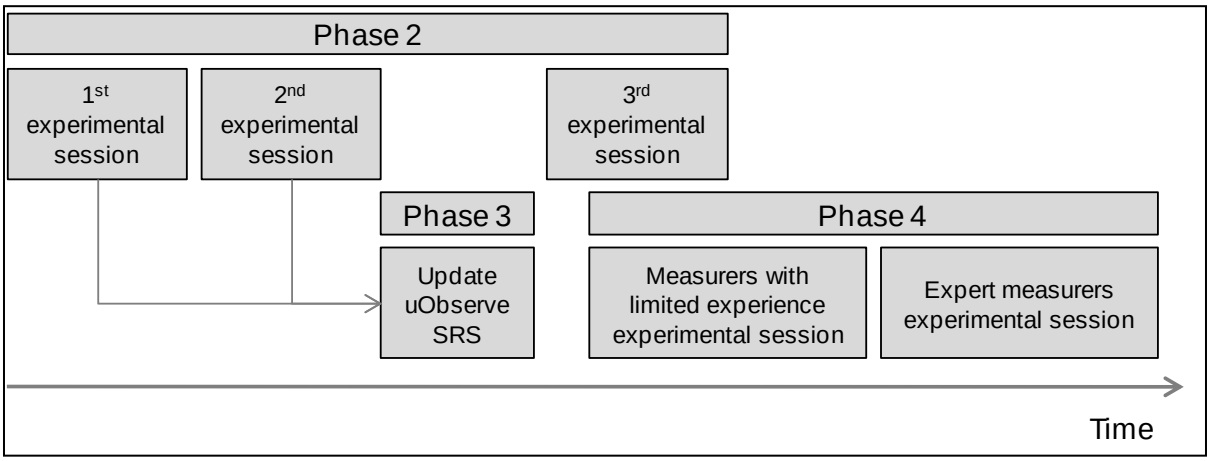

Figure 5.2 Sequence of experimental sessions on a timeline

5.3 Phase 3: Updating the uObserve SRS using identified defects

Phase 4 of this research project required an updated version of the uObserve SRS in order to quantify the impact of defects on the functional size. Therefore, prior to Phase 4 and as part of Phase 3, it was necessary to prepare an updated version of the SRS as experimental material. After completing the first and second experimental sessions from Phase 2, the uObserve SRS was edited by the researcher to fix a majority of defects (78.2%, see Table 5.4) and address most of identified issues (79.5%, see Table 5.5).

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

- Resolved: specific modifications to the SRS solved directly or indirectly the defect or issue.
- Open: the defect or issue has not yet been fixed or addressed.
- Rejected: the defect was rejected during the logging meeting as it was not applicable. As an example, one inspector identified a defect saying ‘No page numbers’ in the bottom of a page, but the SRS pages were numbered on the top of pages, which the inspector did not notice.

When corrections were applied to fix defects, there were a total of 250 unique defects to fix and issues to address (211 defects and 39 issues). Applied modifications were reviewed against the defect list as well as the original version (v1.0) before issuing the new SRS version (v2.0) (see Appendix II). 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 the improved SRS document, knowing it contained residual defects, on the basis that the perfect SRS document might not exist or may require unreasonable effort to obtain.

Table 5.4 Defects fixing status per category and type from uObserve SRS v1.0 to v2.0

Defect status	**Functional**				**Non-functional**				**Total**	**%**
	C	**M**	**S**	**Total**	**C**	**M**	**S**	**Total**		
Resolved	45	56	18	119	15	13	18	46	165	78.2%
Open	13	11	4	28	6	5	7	18	46	21.8%
Total	**58**	**67**	**22**	**147**	**21**	**18**	**25**	**64**	**211**	**100%**

Table 5.5 Issues status per category and type from uObserve SRS v1.0 to v2.0

Issue status	**Functional**		**Non-functional**		**Total**	**%**
	Q	**I**	**Q**	**I**		
Resolved	12	8	2	9	31	79.5%
Open	0	5	0	3	8	20.5%
Total	**12**	**13**	**2**	**12**	**39**	**100%**

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).

5.4 Phase 4: Expert measurers experimental session

5.4.1 Purpose of the expert measurers experiment

The purpose of Phase 4 is to measure the impact on functional size, once the majority of defects were fixed, through an experimental session involving experienced practitioners skilled in measuring functional size with the COSMIC method.

5.4.2 The requirements document

The uObserve SRS v2.0 (see Appendix II) was used in this experiment for Phase 4. An electronic copy of this SRS was sent by email to participants, along with the experiment procedure.

5.4.3 The participants (measurers)

The exact same five COSMIC expert measurers agreed to participate voluntarily in this experiment. For four of them, their participation in the first experiment happened four years before; for one of them, the measuring of the uObserve SRS v1.0 and v2.0 was one year apart.

5.4.4 The experiment steps

1) Prepare the experiment

   A measurement procedure was prepared to ensure that all participants apply the same steps and the same expected effort (Figure 5.3).

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 5.3 Measurement procedure of the expert measurers' experimental session

2) Invite experienced measurers

Expert measurers were invited by email to participate voluntarily in this experimental session.

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 research assistant independently verified that notes from participants were adequately captured in the experiment spreadsheet. Those 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

   FSM and effort data from this experiment were analyzed and compared to data from the previous experts experiment.

5.4.5 Expert measurers experiment data

5.4.5.1 Defects, issues, and assumptions data

Altogether, expert measurers found 11 defects and 25 issues in v2.0 (see Table 5.6), including 20 measurement assumptions. The number of defects is considerably smaller than what was found by expert measurers in v1.0 (86 defects and 11 issues). No duplicate were observed among these items.

Table 5.6 Defects, issues, and assumptions, per category and type, as recorded by measurers

Type	**Defects**			**Issues**			**Total number of elements**
Category	**C**	**M**	**S**	**Q**	**I**	**A**	
Functional	4	5	0	2	2	10	**23**
Non-functional	0	1	1	1	0	10	**13**
Sub-total:	**4**	**6**	**1**	**3**	**2**	**20**	**36**
Total:	**11**			**25**			

5.4.5.2 Functional size data

The uObserve SRS v2.0 had 15 use cases (UC) descriptions with identified triggering events and they were divided into two distinct software applications (the uSpy client and the uSleuth server). 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 impact on their size results but a significant effect on the number of functional processes.

Measurer H, who mentioned not having applied the COSMIC method for at least a year, did not see that the two applications were users of each other, sharing the same human user. Based on the assumption that there was only one single human user for the whole system, he assumed that uSpy client use cases should be ignored when measuring because he considered them as being implementation use cases. This 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, his assumption was declared “wrong” and size and effort data related to v2.0 for this measurer were excluded from calculation of average and standard deviation (see Table 5.7 and Table 5.8).

Table 5.7 Number of functional processes identified

Measurer	**Functional processes**		
	v1.0	**v2.0**	**Difference**
C	14	21	+9
F	10	14	+4
J	10	15	+5
L	14	12	+1
H (see note)	11	8	-3
Average	**11.8**	**15.5**	**3.5**
Std. dev.	**1.8**	**3.4**	**3.4**

Note: v2.0 data excluded from average and standard deviation due to wrong measurement scope assumption during the measurement strategy definition.

Table 5.8 Functional size

Measurer	Functional size (CFP)		
	v1.0	**v2.0**	**Difference**
C	57	81	+24
F	62	71	+9
J	55	97	+46
L	61	68	+7
H (see note)	55	33	-22
Average (excluding H)	**58.0**	**79.3**	**20.5**
Std. dev.	**3.0**	**11.3**	**14.0**

Note: v2.0 size excluded from average and standard deviation due to a wrong measurement scope assumption during the measurement strategy definition.

The average difference of the functional size across SRS versions was 20.5 CFP per measurer.

5.4.5.3 Effort data

Measurers were allotted up to 70 minutes for measuring the functional size of the uObserve SRS v1.0 which contained 10 use case descriptions, but they took less than 60 minutes on average. The uObserve SRS v2.0 contained 15 use cases and measurers were allotted up to 90 minutes to measure its functional size (50% increase from actual measurement effort of SRS v1.0). Table 5.9 shows the effort (in minutes) and the relative effort (in minutes per CFP) spent by each measurer, with the average and standard deviation.

Table 5.9 Absolute and relative effort spent by measurers

Measurer	SRS v1.0			SRS v2.0		
	Effort (minutes)	**Size (CFP)**	**Relative effort (minutes/CFP)**	**Effort (minutes)**	**Size (CFP)**	**Relative effort (minutes/CFP)**
C	75	57	1.3	90	81	1.1
F	49	62	0.8	76	71	1.1
H	45	55	0.8	90	33	2.7
J	60	55	1.1	240	97	2.5
L	60	61	1.0	90	68	1.3
Average	**57.8**	**58.0**	**1.0**	**85.3**	**79.3**	**1.2**
Std dev.	**10.5**	**3.0**	**0.2**	**6.6**	**11.3**	**0.1**

Note: Effort and size from measurer H are excluded from average and standard deviation of v2.0 due to wrong measurement assumption leading to an incomplete sizing. Effort and relative effort from measurer J are excluded from average of v2.0 as an outlier; an additional activity was done by this measurer.

Measurer J reported effort result much 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. When excluding data from measurers J (outlier) and H (incomplete measurement), the average effort to measure v2.0 was 85.3 minutes with a standard deviation of 6.6 minutes. Compared to the effort required to measure v1.0, the new average effort was 48% higher for a size increase of 37%. The relative effort was 20% higher on average with half the standard deviation.

5.4.6 Expert measurer experiment results analysis

The FSM result differences among measurers were all explained through their documented measurement assumptions and identified defects. Except for measurer H who reported an incomplete measurement result, these differences were not measurement errors as developers could have built the software corresponding to any of the proposed sizes. These sizes correspond to different flavours of functional behaviour as the related assumptions could have been made by distinct development teams.

5.5 Limited experience measurers experiment

5.5.1 Purpose and objective of the limited experience measurers experiment

The purpose was to perform an experiment involving software engineers who had recently acquired knowledge in measuring software functional size with the COSMIC method, but with limited practical experience: these measurers had to measure a requirements document, while identifying any defects in the functional requirements.

5.5.2 The requirements document

The uObserve SRS v2.0 was used for this experiment. A printed copy was given to all participants.

5.5.3 The participants (measurers)

Five measurers with limited experience (measurers M, P, Q, R, and S) participated in this experiment.

5.5.4 The experiment steps

1) Prepare the experiment
   Printed copies of uObserve SRS v2.0 were prepared.

2) Invite measurers with limited experience
   A subset of the same measurers with limited experience from the third experimental session participated in this Phase 4 experimental session. They were all registered in the software engineering symposium held at ETS in August 2010. The experimental workshop was part of the symposium program and they all volunteered to participate.

3) Conduct experiment

   Each participant was given a printed copy of the SRS v2.0. The instructions were repeated as they were the same as for the third experimental session: measure the functional size and identify defects using the provided list of rules.

4) 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 identified defects and issues.
   - Detailed FSM results.
   - Effort spent applying the experiment measurement procedure.

5) Objectively ensure accuracy of compiled data

   A research assistant independently verified that written notes from participants were adequately captured in the experiment spreadsheet. Those written notes contained defects, issues, assumptions, and measurement data.

6) Analyze data

   FSM, defect, and effort data from this experiment were analyzed and compared to data from the experts experiment.

5.5.5 Limited experience measurers experiment data

5.5.5.1 Defects, issues, and assumption data

As for the experts, a lower number of defects and measurement assumptions were identified by measurers having limited experience (Table 5.10). There was one critical defect and one minor defect, plus five measurement assumptions.

Table 5.10 Defects and assumptions identified by measurers with limited experience

Measurer	Defects	Assumptions
M	0	0
P	0	3
Q	2	2
R	0	0
S	0	0
Total:	**2**	**5**

The number of identified defects was small (i.e. two defects). There were no duplicate defects as these two defects were found by the same measurer.

5.5.5.2 Functional size data

Functional size measurement data are presented in Table 5.11, including notes related to the quality of measurement results.

Table 5.11 Functional size data from measurers with limited experience

Measurer	# FP	Functional size	Difference with experts' average (79 CFP)	% of difference	Notes
M	12	81	2	3%	3 FP not measured, including the biggest 'Record an experiment'
P	14	87	8	10%	1 FP not measured (the smallest), some superfluous DM
Q	15	130	51	65%	Large number of superfluous DG and DM
R	15	126	47	59%	Large number of superfluous DG and DM
S	5	66	-13	-16%	Only 5 FP measured out of 15, excluded from average
Average:	**14.7**	**114.3**	**35.3**	**45%**	
Std dev.:	**0.5**	**19.4**	**19.4**	**25%**	

5.5.5.3 Effort data

Measurer M took only 30 minutes to measure, saying that this SRS v2.0 was much clearer than the SRS v1.0. However, he did not measure three of the functional processes. The measurement activity was time-boxed to 75 minutes, which were used by all other measurers.

5.5.6 Limited experience measurers: experiment results analyses

5.5.6.1 Challenges applying the COSMIC method

The number of measurers with limited experience who participated in the Phase 4 experimental session was rather small: five measurers only, none of them held a COSMIC entry-level certification. Furthermore, this experiment happened the same day as the third experimental session in which they participated. They did not receive any feedback on their

measurement of uObserve SRS v1.0. Therefore, it was expected that they would face the same measurement challenges with the SRS v2.0. And they did, but not as much.

1) Missing functional processes and related Entry DMs

When looking at the number of identified FP and their related triggering entry, the average improvement was of 0.6 missing Entry DM (from 3.4 to 2.8) (Table 5.12).

Table 5.12 Missing entry movements of triggering events

Measurer	# FP identified	# Triggering Entry	# Missing Entry	% missing from total
M	12	12	0	0%
P	14	7	7	50%
Q	15	13	2	13%
R	15	10	5	33%
S	5	5	0	0%
Average	**12.2**	**9.4**	**2.8**	**19%**
Std dev.	**3.8**	**3.0**	**2.8**	**20%**

But missing complete FP was a bigger challenge. Measurer S seemed to struggle measuring for both SRS versions: he measured only one third of the FPs. His results suggested that his training might not have been completed in order for him to be autonomous in producing FSM results of quality. Measurer M missed three FPs, most probably because he performed his measurement task too fast (only 30 minutes instead of 75) and seemed to have simply skipped the pages on which the missing FP were written.

Measurer P did not measure the ‘Display “About uObserve”’ use case but wrote an assumption for this decision. Thus, it was not a missing FP but a conscious choice that could be explained and discussed. This was an example of a measurement assumption made without the presence of a defect.

Measurers Q and R identified all FPs.

2) Incorrect data group identification and superfluous DMs related to multiple occurrences

Measurers P, Q, R, and S all reported superfluous DM on occurrences. This observation suggested that the related COSMIC rule was not understood. Analyzing how they measured, it appears that they applied a linear measurement process, aligned with every functional step that was described in the SRS: this led them to assign movements to every action, regardless of the previous usage of data groups involved in any single FP.

But the COSMIC method is not a linear process; it is iterative per boundary within a measurement scope, per FP within a boundary, per data group within an FP, and per DM related to a DG within a FP. That concept requires that a measurer identifies data groups by their name. These measurers did not name data groups per se but they used the name of the action including the data group name. It became difficult for them to recognize multiple occurrences as there was confusion between the data group (the object of interest) and the action, or movement, applied to it. Training and training material must emphasize data group naming during the mapping phase to ensure adequate measurement results at the measurement phase.

5.6 The influence of defects on functional size

FSM results from Phase 2 experimental sessions showed an average size of 58.0 CFP as reported by experts. After having fixed 165 defects and resolved 31 issues during Phase 3, FSM results from Phase 4 experiments showed an average size of 79.3 CFP as reported by experts. Having fixed these defects increased the functional size by 36.6% ((79.3-58.0)/58.0). Several defects were related to inadequate functional decomposition and fixing them increased the number of detailed use cases from 10 in the SRS v1.0 to 15 in the SRS v2.0.

Some critical defects were left unresolved in the SRS v2.0, such as the absence of a data model. A defect like this one has inevitably resurfaced, leading to measurement assumptions explaining differences in functional sizing among experts.

There was a smaller standard deviation of functional size from SRS v1.0 (3.0 CFP) to SRS v2.0 (11.3 CFP). FSM and defects results of SRS v1.0 suggested that the significant large number of defects was not favourable to make detailed assumptions, especially related to the high-level defects. This situation may explain why measurers made more assumptions while measuring SRS v2.0 with fewer defects than with SRS v1.0.

CHAPTER 6

DISCUSSION

6.1 Summary of data analysis

Adding a measurer to an inspection team increases the number of identified unique defects and increased the team's efficiency. On the assumption that a measurer would replace an extra inspector, the efficiency also increased, unless the team of inspectors was entirely composed of experts.

Adding a measurer to an inspection team improved the effectiveness of worst teams and was slightly below the effectiveness of median and best teams. The net loss of effectiveness for those teams was -0.3 minute/defect only, which is small considering the teams obtained in addition the functional size of the SRS document.

Measurers with limited experience tend to make a limited number of types of measurement errors, an indicator of a lack of understanding of the COSMIC method.

Once the majority of defects were fixed, the size of the functional requirements in the SRS document increased. The average functional size measured by the experts increased by 36.6%.

6.2 Exploring the measurer participation in an inspection team

Inspections are applied for many reasons, including: A) to find defects for cost savings, and B) to transfer knowledge on a continuous basis by communicating defects to avoid in the future (Gilb and Graham, 1993). When projects adopted inspections to manage their software quality from early life-cycle phases, all team members ended-up participating in inspection teams throughout the project. And projects are typically not staffed with experts only, but staffed with a mix of junior, intermediate, and senior software engineers; similarly reviewer's

skills may greatly vary based on personal abilities to find defects and their experience in participating to inspections. Therefore, having only expert inspectors in an inspection team is not likely to happen often in practice.

Looking at efficiency and unit cost analysis results in section 4.5 suggest that a measurer can bring significant value in improving the efficiency and unit cost of most inspection teams. Improvement results were better as measurers had more experience and skills in applying the COSMIC method. In terms of unit cost, the net gain was bigger with worst teams and below ± 1.4 minute/defects for median and best teams. This means that for a similar cost of adding an extra inspector, an inspection team can add a measurer who will find defects and provide a functional size of the functional requirements being inspected.

6.2.1 The analogy of the chicken and the egg: involving measurers first or last?

Since measurers tend to identify defects at a higher-level of details, a potential involvement of a measurer would be to measure and identify defects before the inspection team does, as an entry criterion to an inspection. However, the initial functional size would have to be measured again after the inspection has been completed, which implies twice the measurement cost.

In contrast, a measurer could be involved only after the inspection is completed. But the measurer is likely to find a significant number of critical and minor defects which would have to be fixed without guarantee of being communicated to the inspectors previously involved. Also, due to the potential richness of defects found by the measurer, involving the measurer after the inspection is not likely to provide skills improvement of inspectors.

6.2.2 Involving the measurer as an inspection team member

When an organization has adopted the practice of inspections, results from this research suggest that adding a measurer to an inspection team would increase the inspection’s

efficiency while not having a significant impact on the inspection's effectiveness. FSM would be measured during Step 3 of the inspection (Inspect the product) and would be reviewed during Step 6 (Re-check with issues) to follow-up with modifications affecting the functional size. In that case, the measurer member of the inspection team would be required to participate in Step 6 in order to ensure the quality of FSM results and maintain measurement costs as low as possible.

6.3 The relationship between defects and functional sizing

In practice, requirements written in a natural language are defect prone and, unless extensively verified, subject to interpretation. Out of 35 participants, 33 found a total of 243 unique confirmed defects in the uObserve SRS v1.0.

On average, measurers have found a larger proportion of high-level defects than inspectors while the majority of measurement assumptions were related to low-level defects.

Since measurement assumptions could explain the variation of functional size measured by experts, these assumptions and variation of size were smaller related to SRS v1.0 than SRS v2.0, mainly due to the presence of a larger number of high-level defects.

If every defect would have had a related measurement assumption, there would have been either a larger variations in the sizing result of SRS v1.0, or a bigger sizing result. This assumption is derived from FSM results of SRS v2.0 where a larger average size was measured with a larger standard variation. In that case, fewer defects were identified and of a lower-level, favouring the definition of measurement assumptions. The impact was an average size increase of 36.6% from SRS v1.0 to SRS v2.0, which is important to a point where it should be required to document these assumptions, whether they would be related to defects or not.

The perfectly written SRS document might very well be a utopia as the natural language in which it would be written is recognized as needing a requirements validation activity (Pressman, 2001, p. 260). Defects in functional requirements are more likely to occur than be inexistent, even after verification and validation activities have been applied to them, as this was the situation with the uObserve SRS v1.0.

Therefore, what should be the behaviour of a measurer facing defects in functional requirements that may have an impact on the resulting functional size? The answer depends on the context in which FSM takes place and the measurement strategy that is agreed upon between the measurer and the FSM results user. When the context requires ignoring defects in functional requirements, the measurer must voluntarily limit the FSM activity to the provided requirements. In that case, it may be a good practice to describe that defects were ignored purposely. In relation to the measurement strategy, many contexts require that defects be raised and their potential solution measured (e.g. estimation, benchmarking, or process improvement oversight). In these cases, raising defects and related assumptions that would be sized seems to be more appropriate.

From this research project results, it is recommended that measurers should indeed identify defects, issues, and assumptions, in order to support FSM results. This additional activity could be added to the COSMIC method, as part of the mapping phase and the measurement phase. SRS documentation, along with its various measurement assumptions on functional descriptions, could lead to different implementation choices and different functional sizes.

6.4 Practical training on the application of the COSMIC method

FSM provides the functional size of a software application, which would allow a software development team or project manager to use this size as an input for estimation, process improvement, and benchmarking. It is therefore important that the measured size resulting from FSM activities be consistent with the artefacts used as input, whether it is measured from requirements or other artefacts.

Most measurers who participated in this research project had limited experience: their knowledge of FSM was mostly theoretical and few of them had actually done size measurement projects, which can partly explain observed measurement differences with expert measurers' FSM results. Going from theory to practice can be inefficient when practical training is not given, particularly on how to apply measurement rules and principles in various cases. Therefore, efficient training sessions should address the identified challenges, supported by meaningful exercises and necessary feedback on their FSM results.

FSM results from limited experience measurers and documents with a large number of functional defects have shown functional size differences of over 25% than sizing results from experts, in 10 cases out of 13. For any organization performing FSM for estimation or benchmarking purposes, it would be recommended that measurement done by a measurer having limited experience be verified by an expert measurer prior to communicate final FSM results, as part as completing their training on applying the COSMIC method. Pairing between an expert and a newly trained measurer could be an efficient way to bring his/her skills at the practitioners' level, similar to a companion-apprentice relationship as it exists in many professions. To ensure the quality of measurement made by measurers with limited experience, the COSMIC trainer would have to include practical exercises using relevant case study material.

6.5 The cost of functional size measurement

As FSM results are primarily used in management activities such as estimating, process improvement oversight, and benchmarking, the measurement activity cost is considered as a management cost for which there is no tangible direct benefit. Applying inspections is considered as a quality assurance (QA) benefit as its cost is generally outweighed by the reduction of rework or testing costs. Therefore, applying FSM within inspections would transfer a management cost (indirect or overhead) to a direct QA benefit. By doing so, the cost of measurement decreases substantially.

When an organization does not apply inspections on its software requirements, every defect identified by a measurer in early phases of the software project becomes a value-added to this project. Software development is a transformation activity: assuring the quality of elements it produces becomes part of the value-chain management (Porter, 1998). Quality related activities are no longer viewed as control element but they are activities bringing value to the software product, assuming that actions are taken on identified defects.

6.6 Threats to validity

6.6.1 Evolutions of the COSMIC method and their impact on this research

The COSMIC method v2.2 was used for the first experimental session involving experts. Within the next year, the method had evolved to v3.0. The main change was that the measurement viewpoint (user or developer) was replaced by a level of granularity which does not limit measurement to two levels but opened the applicability of the COSMIC method to more levels, as deemed appropriate by those applying it. This change did not affect this research since the same requirements document was used, therefore with the same level of granularity on requirement descriptions.

6.6.2 Language barrier of the SRS

Out of the 35 participants to the experiments, only two spoke English as a primary language. All other 33 participants knew English as a secondary language (their primary language spanned from French, Arabic, Dutch, Italian, Spanish, Romanian to Bulgarian). The SRS document was written in English, a language in which measurers were not fluent in – one measurer noted this as a comment on his measurement recording sheet. Misunderstanding of the requirements due to the language barrier may have had an impact on the resulting functional size measurement results. However, that measurer was one of the three measurers whose FSM results were within 20% of experts' average sizing result.

6.6.3 SRS document from only one system used in experiments

This research used two versions of the uObserve SRS document. The system was a hybrid of real-time and management information application of the usability testing business domain, which was not familiar to the majority of participants. Applying the same research protocol on SRS of various system types might have led to different results, more specifically about effectiveness and efficiency.

6.7 Extrapolation of research results

6.7.1 Training on inspection

No known certification exists to qualify inspector's skills. The experience level had an influence of inspectors' ability to identify defects as experts clearly found a larger number of defects (22.8 defects on average) than inspectors with limited experience (5.3 defects on average). Training on the inspection approach was given to participants, which had given them the ability to assign an adequate defect type and category. But an inspector's ability to identify defects, to see what is missing, to relate two groups of information and to uncover their inconsistencies seems to improve with experience and practice as suggested by the number of defects found.

6.7.2 Training of measurers

FSM results from measurers having limited experience might have been closer to experts' results if the practical portion of their training would have led them towards measurement autonomy. Accepting participants in an experiment as measurers as they pretend knowing how to apply the COSMIC method might not be sufficient. None of the limited experience participants from Phase 4 experimental session had a COSMIC Entry-Level certification, and they did face several measurement challenges. More training on the COSMIC method would

have been required to bring their skills at the application level in order to eliminate a number of the measurement challenges observed in this research.

A measurer without any COSMIC certification might be very capable of applying the COSMIC method, but without assessing the measurer' skills, the measurer's capability remains unknown. In order to extrapolate measurement results in the absence of a measurement skills assessment activity, the research protocol should look for COSMIC Entry-Level certified measurers only. However, accepting measurers on their statement that they know the COSMIC method allowed this research project to find COSMIC training improvement opportunities, which might have been different if the experiments participation would have been limited to only those who were certified 'Entry-level'.

6.8 Future research avenues

Assuming that improved practical training would be given to new measurers, experimentation on the efficiency of the improved training would be required to continue improving training material and training means.

Supplemental experiments are required to gather more data on the efficiency and effectiveness of a measurer in finding defects in SRS documents, with or without an inspection team. There is a need to test how sufficient and how efficient would it be to document measurement assumptions in order to explain variation of functional size among measurers.

The uObserve SRS v2.0 needs to be updated with the remaining open defects and be published as a case study that could be used as training material or as measurers' skills assessment material.

CHAPTER 7

RECOMMENDATIONS AND INDUSTRY IMPACTS

7.1 Proposed improvements to the application of the COSMIC method

7.1.1 Adding a measurer's role in peer reviews

This research project has demonstrated that adding a measurer as part of an inspection team allows identifying supplemental defects (i.e. the efficiency increases) with small impact of inspections unit cost. A recommendation from these research results is to add systematically a measurer in an inspection team whose artefact to be reviewed contains functional requirements.

The first advantage would be to find supplemental defects of a different nature than those usually found by inspectors, which would benefit the software project in terms of avoided future rework. The second advantage would be to obtain a functional size that can be used in estimation, status monitoring, process improvement oversight or benchmarking. The third advantage would be to transfer the measurement cost from a management cost to a quality assurance benefit (i.e. a direct project benefit from rework reduction).

7.2 Proposed improvement to the COSMIC measurement manual

7.2.1 Proposed improvement to the COSMIC measurement process

The COSMIC measurement manual v3.0.1 presents the structure of the COSMIC method as a measurement process in three phases (Figure 7.1).

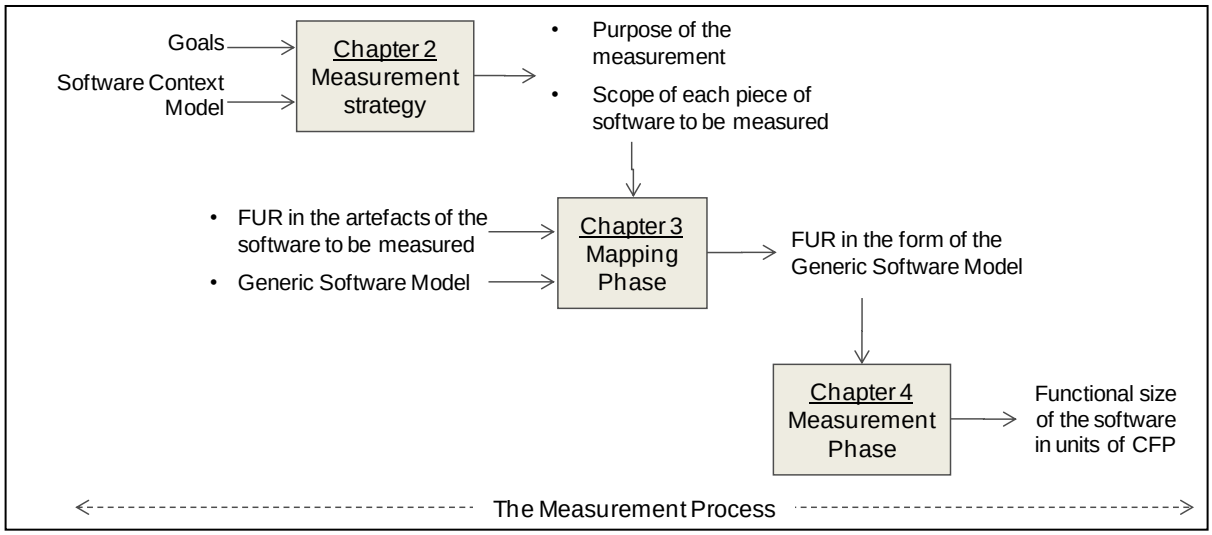

Figure 7.1 Structure of the COSMIC method
Extracted from (Abran *et al.*, 2003)

In order to better describe FSM results, it is recommended that measurers systematically document any identified defects in the artefacts used as inputs to the Mapping Phase. For each defect, at least one measurement assumption has to be made on the potential solution to that defect. For every assumption, the related functional size has to be given at the Measurement Phase. At that point, functional size should be shown as an interval of values where the lowest value corresponds to the size of functional requirements without any assumption on defects solutions, and the highest value includes sizing of defects-related assumptions (Figure 7.2).

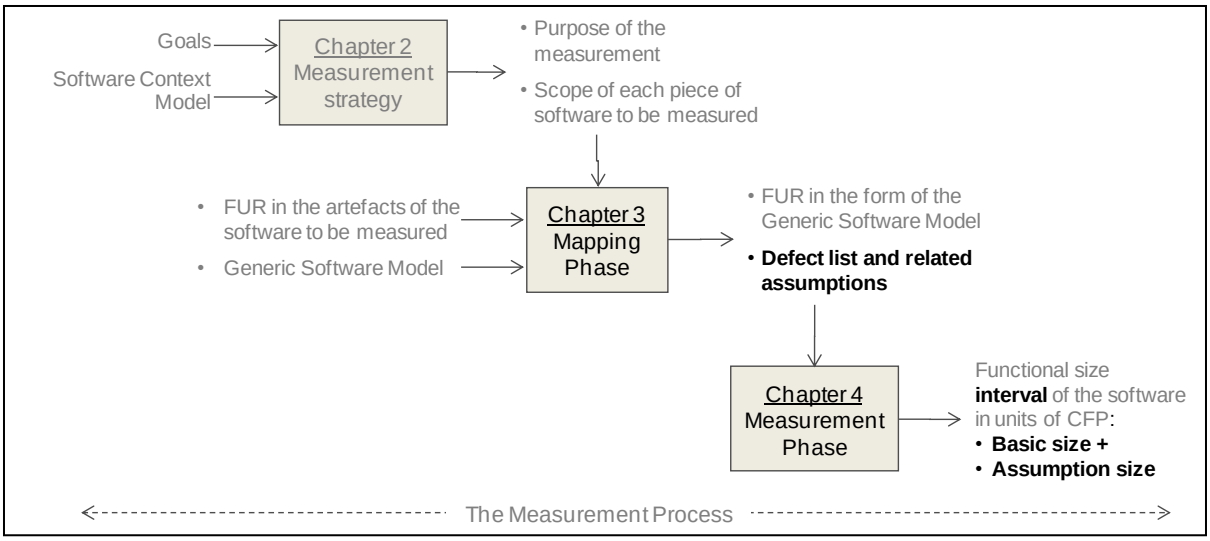

Figure 7.2 Proposed improvement to the structure of the COSMIC method

A verification activity becomes necessary as defects are identified because sizing assumptions need to be confirmed or suppressed. This verification activity could be part of the Measurement Phase (Figure 7.3) or could be shown as a separate phase.

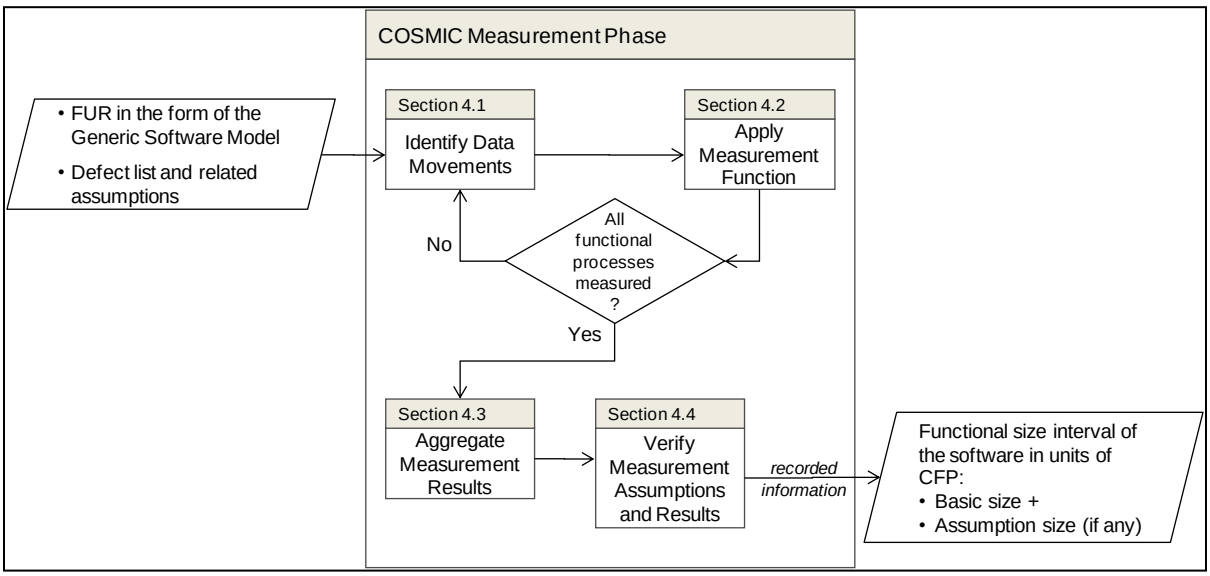

Figure 7.3 Proposed verification activity in the *Measurement Phase*

The objective of the verification activity is to reduce the functional size interval to obtain a more accurate sizing value. This verification of assumptions should take a communication

form (e.g. email, face-to-face meeting, telephone, etc.) between the measurer and a person, or group of persons, having the required skills and knowledge of the software functional behaviour (or expected behaviour) to confirm or reject assumptions. The functional size related to a confirmed assumption is then transferred from the assumption size to the base size. The functional size related to a rejected assumption is then suppressed from the assumption size. The verification activity could go on until all assumptions have been confirmed or rejected, or until the remaining assumptions can no longer be addressed due to missing information (residual assumptions).

7.2.2 Proposed improvement to the measurement labelling rule

Section 5.1 of the COSMIC Measurement Manual describes the rule for COSMIC measurement labelling as follows:

> A COSMIC measurement result shall be noted as "x CFP (v.y)", where:
> - "x" represents the numerical value of the functional size,
> - "v.y" represents the identification of the standard version of the COSMIC method used to obtain the numerical functional size value "x".

As the numerical value of the functional size becomes an interval, it is proposed to modify the rule for COSMIC measurement labelling as follows:

> A COSMIC measurement result shall be noted as "[x z] CFP (v.y)", where:
> - "x" represents the numerical value of the base functional size,
> - "z" represents the numerical value of the assumptions size plus "x",
> - "v.y" represents the identification of the standard version of the COSMIC method used to obtain the numerical functional size value "x".
>
> EXAMPLE: A result obtained using the rules of this Measurement Manual provides a base functional size of 200 CFP while the size of remaining assumptions is 30. The result is recorded as "[200-230] CFP (v3.0)".

7.2.3 Proposed improvement to the measurement reporting rule

Section 5.2 of the COSMIC Measurement Manual describes the rule for COSMIC measurement reporting including 16 attributes that should be recorded. One of these attribute, attribute i), state 'The target or believed error margin of the measurement'. As defects may have a significant impact on the error margin of the measurement, it may also become difficult to estimate that error margin without first identifying defects and their related measurement assumptions. Hence, a recommendation is to add two attributes to this rule, which are described in the following subsections:

- The list of identified defects in the artefacts used as input for sizing
- The list of measurement assumptions that needed to be made, whether or not they are related to a defect.

7.2.3.1 The list of identified defects

While measuring a software size from requirements or other artefacts, a measurer can identify several defects. Examples of defects include missing information that could be partly deducted, wrong or inconsistent information, or ambiguities. Because there is a significant probability that these defects may have an impact on the functional size, it becomes important to provide their list to better serve the purpose of measurement.

The list of defects that a measurer can identify should contain the following attributes and be attached to the COSMIC measurement report:

1) Defect ID.
2) Defect location (e.g. document ID/title, page, chapter, section, paragraph, line).
3) Defect description.
4) Defect type (e.g. critical, minor, spelling).
5) Defect category (e.g. functional, non functional).

A critical defect of the ‘functional’ category should be linked to at least one measurement assumption.

7.2.3.2 The list of measurement assumptions

A measurement assumption corresponds to a candidate solution to an identified defect and should include all that is required to understand the impact of that candidate solution on the functional size:

1) A description of the assumption.
2) The related defect ID.
3) A subset of the COSMIC Generic Software Model and sizing information, including, as minimum:
    a) The functional processes.
    b) The triggering event (when applicable to the assumption).
    c) The data groups.
    d) The related data movements.
    e) The functional size of the assumption.
4) The assumption status (e.g. non-validated, confirmed, or rejected).

When first created, a measurement assumption has a ‘non-validated’ status (i.e. it is a residual assumption) and its size should be reported in the size interval of the measurement labelling. When a measurement assumption is confirmed, its recorded size information should be integrated into the base sizing information and the measurement assumption status should be set to ‘confirmed’. The size of a confirmed measurement assumption should then be excluded from the size of residual assumptions. When a measurement assumption is rejected, its size should be suppressed from the size of residual assumptions.

Several assumptions could be made for the same defect ID where several solutions to the same defect can be proposed. These proposed solutions are likely to be exclusive from one another as no more than one of them should be confirmed. In that case, only the biggest size

among assumptions related to the same defect should be considered in the size of residual assumptions to be reported.

7.2.4 Recording of defects and related measurement assumptions

The Appendix A of the COSMIC Measurement Manual proposes a structure that '*can be used as a repository to hold the results of a measurement for each identified component of an overall scope that has been mapped to the Generic Software Model*' (Abran *et al.*, 2007). That structure is well suited to capture measurement details but leaves no room to identify defects and related measurement assumptions, which is adequate since the user of the list of defects might be different than the user of the functional size.

To improve the COSMIC measurement manual in that matter, it is recommended to add an appendix for the list of defects containing information described in Section 7.2.3.1.

7.3 Research outcomes: Publications of intermediate results and other artefacts

During this research, intermediate results were published and presented in conferences specialized in software measurement, such as the International Workshop on Software Measurement (IWSM)/MENSURA, held annually in different cities around the world. These publications were as follow:

1) Following the first experimental session, whose participants were mainly experts, intermediate results were published at the International Workshop on Software Measurement - IWSM 2008 (Munich, Germany). The paper 'Improving quality of functional requirements by measuring their functional size' contains efficiency and unit cost (effectiveness) analyses between inspectors' results and measurers' results, and provided the value-added or measurers when participating in an inspection (Trudel and Abran, 2008) (see Appendix X). This paper obtained the 'Best paper award' as voted by the conference' participants.

2) Following the second experimental session, whose participants mainly had a limited experience, the data analysis of their common measurement challenges was published at the International Workshop on Software Measurement - IWSM 2009 (Amsterdam, The Netherlands). The paper 'Functional size measurement quality challenges for inexperienced measurers' contains a list of observed challenges that measurers with limited experience were facing while measuring their first software systems (Trudel and Abran, 2009) (see Appendix XI).
3) Using results from the second experimental session, efficiency and unit cost (effectiveness) analyses between inspectors' results and measurers' results was published at the Software Engineering Research, Management & Applications (SERA) conference (Montreal, Canada). The paper 'Functional Requirement Improvements through Size Measurement: A Case Study with Inexperienced Measurers' contains also a comparison of results between expert measurers and measurers with limited experience, as well as a contribution comparison related to the efficiency improvement (Trudel and Abran, 2010b) (see Appendix XII).
4) A more detailed version of the previous paper was published in the International Journal of Computer and Information System (IJCIS): 'The Contribution of Functional Size Measurers in Defects Identification: A Case Study with Inexperienced Measurers' (Trudel and Abran, 2010a) (see Appendix XIII).
5) Results from Phase 4 experimental sessions were published in the International Workshop on Software Measurement - IWSM 2011 (Nara, Japan). The paper 'Bidirectional Influence of Defects and Functional Size' contains the analysis of the influence of defects on the functional size and a recommendation to include documenting defects and related measurement assumptions as part of the measurement activity (Trudel and Abran, 2011) (see Appendix XIV).

7.4 Industry impacts

7.4.1 Contribution to the COSMIC Guideline to ensure measurement verification

Members of the COSMIC Measurement Practice Committee (MPC) used papers published in 2008 and 2009 as intermediate results of this research (see Appendices X and XI) as inputs to create the 'Guideline for assuring the accuracy of measurements' (Desharnais, Lesterhuis and Symons, 2011). These research results contributed to portions of the error-prevention and defect-detection chapters, namely in relation to the quality of measurers as well as the quality of artefacts used as input for the measurement process.

7.4.2 Measurement checklist to avoid common measurement errors

Measurement results from the participants with limited experience with the COSMIC method suggested that improved training material would be required. A one-pager 'COSMIC Quick Reference Card' as illustrated in Figure 7.4 is proposed as a checklist for applying the COSMIC method This reference card aims to improve the quality of the measurement results of practitioners with limited experience. It also includes quality checklist items from the 'Guideline for assuring the accuracy of measurements'.

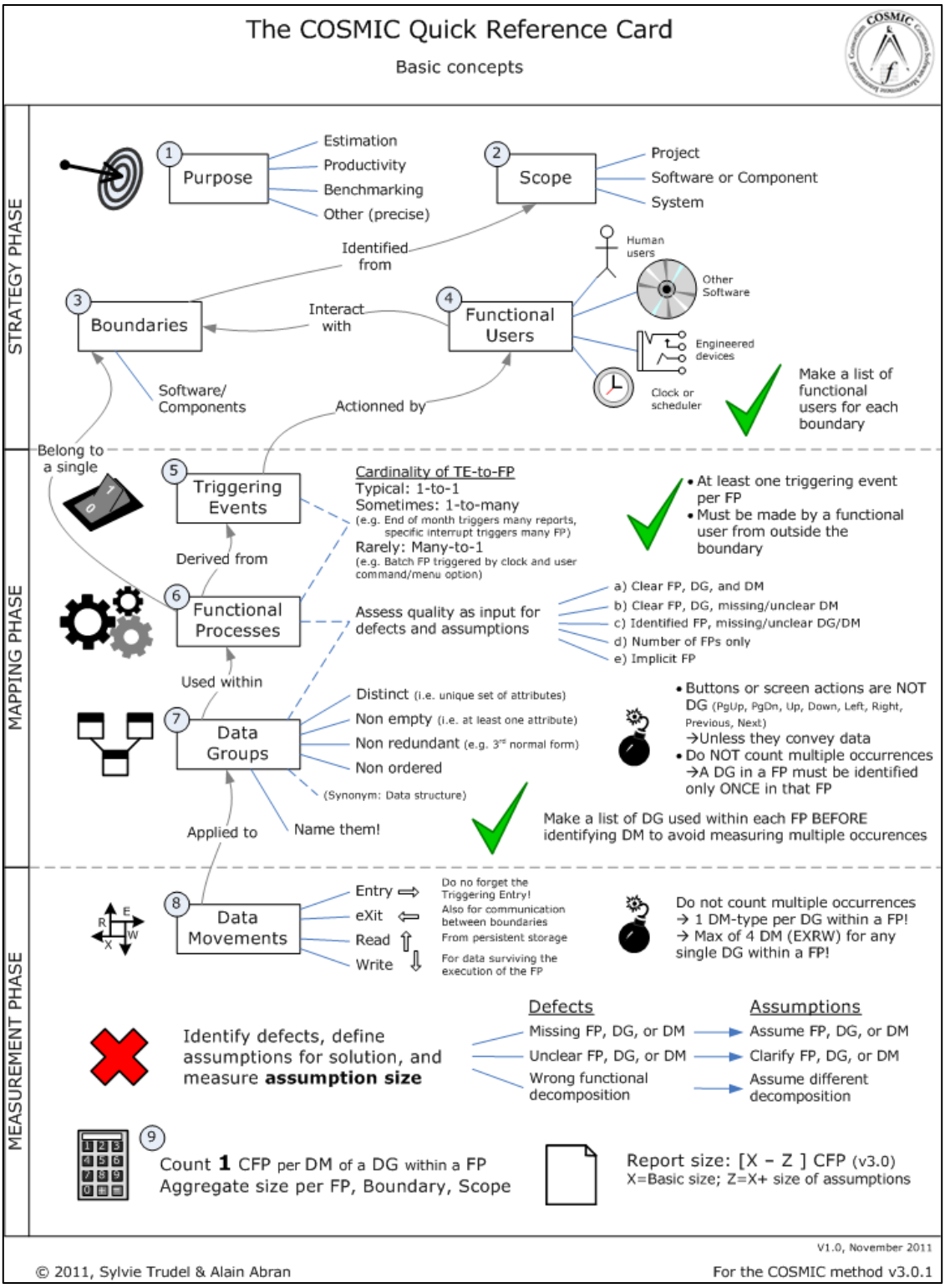

Figure 7.4 A COSMIC Quick Reference Card

7.5 Proposed improvements related to practical training

7.5.1 Include practical exercises in the COSMIC practitioners' training

Measurement results from participants with limited experience suggested the need to systematically include several measurement exercises as part of the basic practitioners' training. The uObserve SRS v2.0, which is reputed containing fewer defects than its v1.0, could be used as one of the measurement practical exercises. Because it contains two software boundaries with several functional processes in each boundary, the uObserve SRS v2.0 would allow a complete practical measurement exercise that could be done within 60 to 90 minutes.

To guide new practitioners and complete the exercise, a list of guiding questions and instructions should be given to the COSMIC training participants, such as:

1) Read the SRS document at least once before answering the questions.
2) What is the scope of this SRS? Can you name that scope?
3) What are the software boundaries in this SRS?
4) For each identified boundary, identify all of its functional users. (Hint: a piece of software within a boundary may be a functional user of another piece of software in another boundary),
5) Within each boundary, identify triggering events and their related functional processes.
6) For each functional process, identify the list of unique data groups being manipulated. (Hint: do not forget the triggering event and do not identify a data group more than once per functional process).
7) For all identified data groups within each functional process, identify all relevant data movements (Entry, eXit, Read, Write). (Hint: there is always an Entry associated to the Triggering event).
8) Count one CFP per DM. Aggregate the functional size per functional process, per boundary, and for the whole measurement scope.

7.5.2 Using the uObserve SRS v1.0, v2.0 or v3.0 to assess measurers' skills

It is planned to fix the remaining open defects of uObserve v1.0 and v2.0 and to release v3.0 with, hopefully, fewer defects. The three different versions could be used as training material to assess measurers' skills as follows:

- uObserve SRS v3.0: should contain few defects, be the easiest version to measure in order to target beginners.
- uObserve SRS v2.0: contains several remaining defects and should be used with intermediate practitioners in order to assess their ability to measure while identifying defects and defining measurement assumptions.
- uObserve SRS v1.0: contains many defects and should be used with experts practitioners in order to assess their ability to measure while identifying defects and defining measurement assumptions.

7.5.3 Using the uObserve SRS v1.0 to assess inspectors' skills

This research project has cumulated over 350 defects and issues on the uObserve SRS v1.0 document. This particular version could be used to assess inspectors' skills and measure their personal or teams' efficiency in finding defects. New defects could be added to the defects repository.

In April 2011, Zayed University in Abu Dhabi held a student peer review contest using the uObserve SRS v1.0 (see Appendix XV). The competition chair, Dr. Manar Abu Talib, was one of this research participants and she asked the researcher and the co-author the permission to use the SRS for that matter.

ANNEX I

LIST OF APPENDICES

The following appendices are referenced as artefacts within this thesis and are included on an attached CD-ROM.

App. #	File name	Content
I	uObserve_Specs_Eng-v1.0.doc	uObserve SRS v1.0
II	uObserve_Specs_Eng-v2.0.docx	uObserve SRS v2.0
III	Inspection_Workshop_slides_Eng .pdf	Presentation slides from inspection training material
IV	Inspection_summary_Eng_v3.pdf	Summary of CRIM's inspection method
V	Inspection_detailed_Eng_v3.pdf	CRIM's inspection method in details
VI	Inspection_form_Eng_v3.pdf	Inspection form
VII	Peer_Reviews_Rules.doc	Peer review rules
VIII	Peer_Review_Roles.doc	Peer review roles
IX	DGL_ProjetPilote_Data.xls	Data from pilot project
X	IWSM2008_Strudel_AAbran.pdf	Paper published in IWSM 2008 proceedings, entitled "Improving quality of functional requirements by measuring their functional size"
XI	IWSM2009_Strudel_AAbran.pdf	Paper published in IWSM 2009 proceedings, entitled "Functional size measurement quality challenges for inexperienced measurers"
XII	SERA2010_STrudel-AAbran.pdf	Paper published in SERA 2010 proceedings, entitled "Functional requirements improvements through size measurement: a case study with inexperienced measurers"

App. #	File name	Content
XIII	IJCIS2010_STrudel_AAbran.pdf	Paper published in IJCIS, entitled "The contribution of functional size measurers in defect identification: a case study with inexperienced measurers"
XIV	IWSM2011_Strudel_AAbran.pdf	Paper accepted for the proceedings of IWSM 2011, entitled "Bidirectional influence of defects and functional size"
XV	Zayed-IT-Competition.pdf	Certificate for participation and contribution for providing free usage of the uObserve SRS v1.0 in a student peer review contest held in spring 2011 at Zayed University.

LIST OF BIBLIOGRAPHICAL REFERENCES

Abran, Alain, Jean-Marc Desharnais, Serge Oligny, Denis St-Pierre and Charles Symons. 2003. *The COSMIC Functional Size Measurement Method Version 2.2 – Measurement Manual – (The COSMIC Implementation Guide for ISO/IEC 19761: 2003)*. Montreal: The Common Software Measurement International Consortium (COSMIC), 81 p.

Abran, Alain, Jean-Marc Desharnais, Serge Oligny, Denis St-Pierre and Charles Symons. 2007. *The COSMIC Functional Size Measurement Method Version 3.0 – Measurement Manual – (The COSMIC Implementation Guide for ISO/IEC 19761: 2003)*. Montreal: The Common Software Measurement International Consortium (COSMIC), 80 p.

Abran, Alain, James W. Moore, Pierre Bourque and Robert Dupuis. 2002. *Software Engineering Body of Knowledge*. IEEE Computer Society.

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

Baraby, Bernard. 2006. « Pistes de solutions pour augmenter la qualité des cas d'utilisation en entreprise ». Projet de maîtrise, Montréal, École de Technologie Supérieure, 88 p.

Basili, V.R., R.W. Selby and D.H. Hutchens. 1986. « Experimentation in software engineering ». *IEEE Transactions on Software Engineering*, n° SE-9, p. pp. 733-743

Beck, Kent. 2000. *Extreme programming explained embrace change*. Boston: Addison-Wesley, 190 p.

Boehm, Barry W. 1981. *Software Engineering Economics*. Englewood Cliff, New Jersey: Prentice-Hall, 767 p.

Chen, Peter. 1977. *The Entity-Relationship Approach to Logical Database Design*. QED Information Systems.

Chrissis, Mary Beth, Mike Konrad and Sandy Schrum. 2003. *CMMI: Guidelines for process integration and product improvement*. Coll. « SEI Series in Software Engineering ». Boston: Addison-Wesley.

DeMarco, Tom. 1979. *Structured Analysis and System Design*. Prentice-Hall.

Desharnais, Jean-Marc. 2003. « Application de la mesure de taille fonctionnelle COSMIC-FFP: Une approche cognitive ». Montréal, UQÀM, 351 p.

Desharnais, Jean-Marc, Arlan Lesterhuis and Charles Symons. 2011. *Guideline for assuring the accuracy of measurements*. 21 p.

DND. 1997-2000. « The software inspection method ». St-Jean-sur-Richelieu: The Canadian Department of National Defence (DND), Developed for the ADATS program (Oerlikon Aerospace).

DoD. 1994. *Defense System Software Development*. DOD-STD-2167A. ANSI.

DoD. 1998. *Software Development and Documentation*. Mil-Std-498. ANSI.

Ducharme, Sylvain, and Sylvie Trudel. 2004. *Analyse des besoins et gestion des exigences de logiciels*. Montréal: CRIM.

Fagan, M. E. 1976. « Design and code inspections to reduce errors in program development ». *IBM Systems Journal*, vol. 15, n° 3, p. 182-211

Gane, T., and C. Sarson. 1982. *Structured System Analysis*. McDonnell Douglas.

Gilb, Tom, and Dorothy Graham. 1993. *Software Inspections* (31 December 1993). Addison-Wesley Professional, 496 p.

IEEE. 1998a. *IEEE Guide for Information Technology - System Definition - Concept of Operations (ConOps) Document*. IEEE Std 1362-1998. New York, NY: The Institute of Electrical and Electronics Engineers, 21 p.

IEEE. 1998b. *IEEE Recommended Practice for Software Requirements Specifications*. IEEE Std 830-1998. New York, NY: The Institute of Electrical and Electronics Engineers, 31 p.

ISO. 2002a. *Functional size measurement -- Part 4: Reference model*. ISO/IEC 14143-4. Geneva, Switzerland: International Organization for Standardization.

ISO. 2002b. *Software engineering -- Software measurement process*. ISO/IEC FDIS 15939. Geneva, Switzerland: International Organization for Standardization, 37 p.

ISO. 2002c. *Software engineering – Mk II Function point analysis – Counting practices manual*. ISO/IEC 20968. Geneva, Switzerland: International Organization for Standardization.

ISO. 2003a. *Functional size measurement -- Part 3: Verification of functional size measurement methods*. ISO/IEC 14143-3. Geneva, Switzerland: International Organization for Standardization.

ISO. 2003b. *Software engineering -- COSMIC-FFP -- A functional size measurement method*. ISO/IEC 19761. Geneva, Switzerland: International Organization for Standardization.

ISO. 2004. *Functional size measurement -- Part 5: Determination of functional domains for use with functional size measurement*. ISO/IEC 14143-5. Geneva, Switzerland: International Organization for Standardization.

ISO. 2005. *Software engineering -- NESMA functional size measurement method version 2.1 -- Definitions and counting guidelines for the application of Function Point Analysis*. ISO/IEC 24570. Geneva, Switerland: International Organization for Standardization.

ISO. 2006. *Functional size measurement -- Part 6: Guide for use of ISO/IEC 14143 series and related International Standards* ISO/IEC 14143-6. Geneva, Switzerland: International Organization for Standardization.

ISO. 2007. *Functional size measurement -- Part 1: Definition of concepts*. ISO/IEC 14143-1. Geneva, Switzerland: International Organization for Standardization.

ISO. 2008. *Software life cycle processes*. ISO/IEC 12207. Geneva, Switzerland: International Organization for Standardization.

ISO. 2009. *Software and systems engineering -- Software measurement -- IFPUG functional size measurement method 2009*. ISO/IEC 20926. Genève, Suisse: International Organization for Standardization.

ISO. 2010. *Systems and software engineering -- FiSMA 1.1 functional size measurement method*. ISO/IEC 29881. Geneva, Switzerland: International Organization for Standardization.

ISO. 2011a. *Functional size measurement -- Part 2: Conformity evaluation of software size measurement methods to ISO/IEC 14143-1*. ISO/IEC 14143-2. Geneva, Switzerland: International Organization for Standardization.

ISO. 2011b. *Software engineering -- COSMIC: A functional size measurement method*. ISO/IEC 19761. Geneva, Switzerland: International Organization for Standardization.

Jacobson, Ivar, Grady Booch and James Rumbaugh. 1999. *The unified software development process*. Coll. « Addison-Wesley object technology series ». Reading, Mass. ; Don Mills, Ont.: Addison-Wesley, xxix, 463 p.

Jones, Capers. 1996. *Applied Software Measurement*, 2nd. New York, NY: McGraw-Hill.

Kotonya, Gerald, and Ian Sommerville. 1998. *Requirements engineering: Processes and techniques*. Chichester, England: Wiley.

Leffingwell, Dean, and Don Widrig. 2003. *Managing software requirements : a use case approach*, 2nd. Coll. « Addison-Wesley object technology series ». Boston: Addison-Wesley, xxxvii, 502 p.

Lesterhuis, Arlan, and Charles Symons. 2007. *The COSMIC Functional Size Measurement Method Version 3.0 -- Method Overview*. The COSMIC Group, 25 p.

Lokan, Christopher J. 2004. *Function Points*. Canberra, Australia: School of Information Technology and Electrical Engineering, 45 p.

Nagano, Shin-ichi, and Tuneo Ajisaka. 2005. « Improvement of analysis model by removing improper parts based on functional size measurement ». In *Innovations in Software Measurement, 15th International Workshop on Software Measurement* (September 12-14), Shaker-Verlag. p. 241-254. Montreal, Canada: Otto-Von-Guericke-Universität Magdeburg.

Nishiyama, Shigeru, and Tsuneo Furuyama. 1994. « The validity and applicability of function point analysis -- as related to specification quality and ergonomics -- ». In *EOQ-SC'94*. p. 479-490. Basel, Switzerland.

Paulk, Mark C., Bill Curtis, Mary Beth Chrissis and Charles V. Weber. 1993. *Capability Maturity Model for Software, version 1.1*. Pittsburg, PA: Carnegie Mellon University.

Porter, Michael E. 1998. *Competitive advantage: Creating and sustaining superior performance* (June 1998). Free Press, 592 p.

Pressman, Roger S. 2001. *Software Engineering - A practitioner's approach*, 5th. New York, NY: McGraw-Hill, 860 p.

Sawyer, Pete, and Gerald Kotonya. 2002. « Software Requirements ». In *Software Engineering Body of Knowledge*, IEEE Computer Society.

Sommerville, Ian. 2004. *Software engineering*, 7th. Boston, Mass.: Pearson/Addison-Wesley, xxii, 759 p. <http://www.loc.gov/catdir/toc/ecip0416/2004007038.html >.

Stewart, R., and L. Priven. 2008. « Revitalizing Software Inspections ». In *Software Process Improvement Network (SPIN)* (February 6th, 2008). Montreal, Canada.

Thayer, Richard H., and M. Dorfman. 2000. *Software requirements engineering*, 2nd. Los Alamitos, CA: IEEE Computer Society Press, Institute of Electrical and Electronics Engineers, xvii, 531 p.

Trudel, Sylvie. 2002. « Les inspections logicielles ». In *SPIN de Montréal* (26 March 2002), CRIM. Montréal.

Trudel, Sylvie. 2003. « Atelier: Les Inspections Logicielles ». In *Concordia University Software Engineering Conference (CUSEC)* (January 17th, 2003). Montreal. <http://www.cusec.net/archives/2003/trudel.pdf>. Consulted on April 6th, 2012.

Trudel, Sylvie. 2007. *Software Inspections Workshop*. Montreal, Canada: CRIM.

Trudel, Sylvie, and Alain Abran. 2008. « Improving quality of functional requirements by measuring their functional size ». In *IWSM 2008, MetriKon 2008, ans Mensura 2008* (November 18-19, 2008). p. 287-301. Munich, Germany: Springer-Verlag.

Trudel, Sylvie, and Alain Abran. 2009. « Functional size measurement quality challenges for inexperienced measurers ». In *IWSM 2009 and Mensura 2009* (November 4-6, 2009). p. 157-169. Amsterdam, The Netherlands: Springer-Verlag.

Trudel, Sylvie, and Alain Abran. 2010a. « The Contribution of Functional Size Measurers in Defects Identification: A Case Study with Inexperienced Measurers ». *International Journal of Computer and Information System – IJCIS*, vol. 11, n° 3 (September 2010), <http://www.acisinternational.org/journal/v11n3.html>.

Trudel, Sylvie, and Alain Abran. 2010b. « Functional Requirement Improvements through Size Measurement: A Case Study with Inexperienced Measurers ». In *Software Engineering Research, Management & Applications (SERA)* (May 24-26, 2010). p. 181-189. Montral, Canada: IEEE Computer Society.

Trudel, Sylvie, and Alain Abran. 2011. « Bidirectional Influence of Defects and Functional Size ». In *International Workshop on Software Measurement and International Conference on Software Process and Product Measurement (IWSM-MENSURA) 2011* (November 3-4, 2011). p. 69-75. Nara, Japan: IEEE Computer Society.

Weigers, Karl E. 2002. *Peer reviews in software: a practical guide*. Boston, MA: Addison-Wesley.

Weigers, Karl E. 2003. *Software requirements*, 2nd. Redmond, Wash.: Microsoft Press, xii, 350 p.

Yourdon, Edward N., and L. L. Constantine. 1978. *Structured Design*. Yourdon Press.