ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THESIS PRESENTED TO
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

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

BY
Khaled ALMAKADMEH

DEVELOPMENT OF A SCALING FACTORS FRAMEWORK TO IMPROVE THE APPROXIMATION OF SOFTWARE FUNCTIONAL SIZE WITH COSMIC - ISO19761

MONTREAL, JUNE 12, 2013

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Alain Abran, Thesis Supervisor
Department of Software Engineering & Information Technologies at École de technologie supérieure

Mr. Michaël Gardoni, President of the Board of Examiners
Department of Automated Production Engineering at École de technologie supérieure

Mr. Abdelouahed Gherbi, Member of the jury
Department of Software Engineering & Information Technologies at École de technologie supérieure

Mr. Hamid Mcheick, External Evaluator
Department of Computer Sciences and Mathematics at Université du Québec à Chicoutimi

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON JUNE 11, 2013

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGMENTS

I would like to express my gratitude to all who have supported me in my doctoral study. First and foremost: my research director Professor Alain Abran, for his incredible support in all stages of my doctoral study. His kindness, encouragement, wise advice, knowledge, and professionalism have made working with him a great pleasure and honor. His professionalism and positive attitude towards scientific research have encouraged me to work day and night to finish my doctoral study within three years. He nicely devoted much of his time in face to face meetings and fast email feedbacks to help me pursue my research project. I feel that he is always with me to guide me to the correct path, like a torch that guides its holder through his/her way. I believe that a doctoral student like me would not find such an extra-ordinary research director other than professor Abran. Also, I would like to thank everyone in the software engineering research laboratory and the department of software engineering and information technologies at École de technologie supérieure for their support.

I would like to thank the Hashemite University in the Hashemite Kingdom of Jordan for granting me three years doctoral scholarship to pursue my doctoral studies, in particular Dr. Kamal Bani-Hani the President of the Hashemite University.

I am deeply grateful to the jury members; Professor Mickaël Gardoni, Professor Abdelouahed Gherbi, and Professor Hamid Mcheick for taking time to review my doctoral thesis.

A great love to my parents, my father (Abdelkarim) the source of strength and wisdom in my whole life, my mother (Hanan), in which I want to spend my entire life to thank and honor. You have shown me the joy of intellectual pursuit ever since I was a child. Thanks to my brothers (Mhammed, Ahmad, Ali, and Alaa) and my sister (Shaima). And, I will not forget to thank my uncle (Professor Ibrahim) who has always supported and encouraged me throughout my whole life, my deep appreciation to all my friends in Jordan (Dr. Malik

Qasaimeh, Dr. Khalid Al-Sarayreh, Dr. Kenza Meridji, Dr. Ahmad Qattoum). Last but not least, I would like to thank his highness Mr. Basheer Zoubi and his highness Mr. Adli Al-Khaledi: the Ambassador and the First Secretary of Jordan in Ottawa (Canada) for their encouragement and endless support.

DEVELOPMENT OF A SCALING FACTORS FRAMEWORK TO IMPROVE THE APPROXIMATION OF SOFTWARE FUNCTIONAL SIZE WITH COSMIC - ISO19761

Khaled ALMAKADMEH

RÉSUMÉ

De nombreuses organisations de développement de logiciels s'efforcent de fournir des produits de haute qualité tout en gardant un équilibre entre la satisfaction du client, le calendrier et le budget. L'estimation de l'effort de développement des projets logiciel est l'un des défis majeurs de ces organisations de développement et ce défi est généralement rencontré dès les premières phases du cycle de vie du développement.

Pour relever ce défi, les organisations de développement de logiciels utilisent des techniques d'estimation précoce pour obtenir des estimations de l'effort au début (c.-à-d. estimations *a priori*) afin d'aider les gestionnaires de projet et les responsables techniques dans la planification et la gestion des projets.

L'une des approches pour l'estimation de l'effort *a priori* est basée sur l'approximation des fonctions attendues du logiciel. Ceci nécessite l'utilisation d'une méthode de mesure pour quantifier ces fonctions: la littérature réfère à la mesure de la taille fonctionnelle des produits logiciels - incluant les applications d'entreprise. Différentes normes internationales ont été adoptées pour mesurer la taille fonctionnelle des logiciels, telle que ISO 19761: COSMIC.

Cependant, durant les premières phases du cycle de vie du développement logiciel, et plus spécifiquement dans le processus d'estimation de la taille fonctionnelle du logiciel, l'absence de spécifications complètes et détaillées des exigences logicielles est commune, ce qui entraîne de nombreux défis. Par exemple: le niveau de granularité (c.-à-d. le niveau de détail) de la spécification des exigences fonctionnelles du logiciel est identifié subjectivement en utilisant l'intuition, l'expérience et/ou les opinions des experts du domaine; les facteurs d'échelle ne sont pas attribués; il n'y a pas une notation standardisée pour définir un ensemble standard de facteurs d'échelle que les ingénieurs des exigences peuvent affecter aux spécifications des exigences fonctionnelles des nouveaux projets de développement de logiciels afin d'identifier leur niveau de granularité.

Ces défis affectent l'estimation de la taille fonctionnelle de nouveaux projets de développement de logiciels puisque le résultat de l'estimation de la taille fonctionnelle est l'une des entrées principales du processus d'estimation d'effort. Ces défis empêchent les gestionnaires des projets logiciels de construire des modèles réalistes d'estimation de l'effort pour les nouveaux projets de développement de logiciels.

La motivation de ce projet de recherche est d'aider les organisations du développement logiciels et, en particulier, les gestionnaires des projets et les responsables techniques pour construire des modèles d'estimation de l'effort plus précis et ce en améliorant l'une des entrées du processus d'estimation de l'effort, afin d'améliorer la planification, la gestion et le développement des logiciels à des phases précoces du cycle de vie du développement des logiciels.

Le but de ce projet de recherche est d'améliorer l'une des entrées du processus d'estimation de l'effort et en particulier la qualité de l'approximation de la taille fonctionnelle des nouveaux projets du développement des logiciels.

L'objectif principal de la recherche est de concevoir un cadre de référence à être utilisé par les ingénieurs des exigences pour attribuer des facteurs d'échelle pour les premières versions de la spécification des exigences fonctionnelles du logiciel afin d'identifier leur niveau de granularité, ce qui se déroule généralement après l'étape de l'étude de faisabilité pour les nouveaux projets du développement logiciels.

Pour atteindre cet objectif de recherche, les principales phases de la méthodologie de recherche sont:

- la phase de recherche exploratoire: pour d'étudier l'impact du problème de recherche sur l'approximation de la taille fonctionnelle;
- la phase de conception du cadre de référence: pour concevoir la cadre de référence qui attribue les facteurs d'échelle à des spécifications fonctionnelles des exigences fonctionnelles pour identifier leurs niveaux de granularité; et
- la phase de vérification du cadre de référence: c'est la phase qui vérifie la convivialité du cadre de référence grâce aux différents groupes de participants ayant des profils d'expérience différents, et qui vérifie l'applicabilité de cadre de référence avec une variété d'études de cas représentant des systèmes logiciels différents.

Le principal résultat de ce projet de recherche est un cadre de référence qui se compose:

- d'un méta-modèle qui identifie les concepts et leurs relations qui doivent être recueillies par les ingénieurs des exigences pour atteindre la pleine spécification fonctionnelle des spécifications des exigences logicielles; et
- les critères qui permettent d'identifier le niveau de granularité de la spécification des exigences logicielles, et de leur attribuer des facteurs d'échelle pour classer leurs niveaux de granularité.

Le cadre de référence a été vérifié pour utilisation avec la même étude de cas par trois groupes de participants de l'industrie du génie logiciel, tandis que son applicabilité a été vérifiée avec quatre études de cas.

Mots-clés: facteurs d'échelle, taille fonctionnelle, estimation de taille, exigences incomplètes, projets logiciels, ISO19761.

DEVELOPMENT OF A SCALING FACTORS FRAMEWORK TO IMPROVE THE APPROXIMATION OF SOFTWARE FUNCTIONAL SIZE WITH COSMIC - ISO19761

Khaled ALMAKADMEH

ABSTRACT

Many software development organizations strive to deliver high-quality products while keeping a balance between customer satisfaction, time, and budget. The estimation of the effort of software development projects is one of the major challenges of these software development organizations. This challenge is typically faced at early phases of the software development life cycle.

To tackle this challenge, the software development organizations use early estimation techniques to obtain early effort estimates (i.e. *a priori* estimates) in order to help project managers and technical leaders in projects planning and management.

One of the methodologies for *a priori* effort estimation is based on the approximation of the expected software functionality. This requires the use of a measurement method to quantify this functionality: the literature refers to the measurement of the functional size of software products – including business applications. Various international standards have been adopted to measure the functional size of software such as ISO 19761: COSMIC.

However, during the early phases of the software development life cycle, and more specifically in the approximation of the functional size of the software expected to be developed, the lack of detailed and complete software requirements specifications is common, which leads to many challenges. For instance, the level of granularity (i.e. the level of details) of the functional requirements specifications of software is identified subjectively using intuition, experience and/or opinions of the field experts. Also, there is no standardized notation to define a standard set of scaling factors to be assigned by the requirements engineers to the functional requirements specifications of software projects to identify theirs levels of granularity.

These challenges affect the quality of the functional size approximation of software development projects, since the result of the functional size approximation process is one of the primary inputs for the a priori effort estimation process. These challenges prevent the estimators of software projects from building realistic effort estimation models.

The motivation of this research project is to help software organizations and in particular projects managers and technical leaders to build more accurate effort estimation models by improving one of the inputs for the effort estimation process, in order to improve the planning, the management, and the development of software at early phases of the software development life cycle.

The goal of this research project is to improve one of the inputs of the *a priori* effort estimation process, and in particular the functional size approximation of software development projects.

The main research objective is to design a framework – to be used by the requirements engineers – that assigns scaling factors to early versions of functional requirements specifications of software to identify their levels of granularity at the early stages of the software development life cycle.

To achieve this research objective, the main phases of the research methodology are:

- exploratory research: to investigate the impact of the research issue on the approximation of the functional size approximation process;
- framework design: to design the framework that assigns scaling factors to functional requirements specifications to identify their levels of granularity; and
- framework verification: to verify the usability of the framework by different groups of participants with different experience profiles, and to verify the applicability of the framework with a variety of case studies representing different software systems.

The main outcome of this research project is a framework that consists of: a meta-model that identifies the relevant concepts and the relationships that need to be collected by the requirements engineers for achieving full functional specification of software requirements specifications, as well as criteria that identify the levels of granularity of software requirements specifications, and assign scaling factors to rank their levels of granularity.

This framework is verified for usability with the same case study by three groups of practitioners in the software engineering industry and verified next for applicability with four case studies.

Keywords: scaling factors, functional size, size approximation, incomplete requirements, software projects, ISO19761.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

AFM	Application Feature model
AHP	Analytic Hierarchy Process
BFC	Base Functional Component
CÉR	ÉTS Committee for Ethics in Research
CFP	Cosmic Function Point
COSMIC	Common Software Measurement International Consortium
CRUD	Create, Read, Update and Delete
DFM	Domain Feature Model
EI	External Input
EIF	External Interface File
EMOF	Extended Meta Object Facility
EO	External Output
EQ	External Query
E&Q	Early & Quick
ÉTS	École de Technologie Supérieure
FP	Functional Process
FP	Function Point
FPA	Function Point Analysis
FSM	Functional Size Measurement
FUR	Functional User Requirement
GC	Generic Criterion
GP	General Process

GUI	Graphical User Interface
HAT	Hierarchy Action Tree
IDE	Interactive Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IFPUG	International Function Point Users Group
ILF	Internal Logical File
IS	Information System
ISBSG	International Software Benchmarking Standard Group
ISO	International Organization for Standardization
ISSEM	International Symposium for Software Engineering Management
IT	Information Technology
IWSM	International Workshop on Software Measurement
JTC	Joint Technical Committee
KAs	knowledge Areas
KPA	Key Process Area
LESIA	Laboratoire d'Environnements de Synthèse et Interfaces Avancées
MDA	Model Driven Architecture
MDD	Model Driven Development
MP	Macro Process
NFR	Non-Functional Requirement
OASIS	Open Artwork System Interchange Standard
OCL	Object Constraint Language

OMG	Object Management Group
OO	Object Oriented
ROOM	Real-time Object Oriented Modeling
RUP	Rational Unified Process
SC	Specified Criterion
SD	Strategic Dependence
SDLC	Software Development Life Cycle
SFSU	Scenario Functional Size Unit
SR	Strategic Rationale
SRS	Software Requirements Specifications
Std	Standard
SWEBOK	Software Engineering Body of Knowledge
TP	Typical Process
UCP	Use Case Point
UFSU	Use-case Functional Size Unit
UML	Unified Modeling Language
xUML	Executable UML

INTRODUCTION

This thesis reports on the research carried out to improve the approximation of the functional size of early requirements specifications of software development projects with the aim to help the managers of these software projects in building more realistic effort estimation models using the available tools and methodologies. The improvement will be achieved by improving the primary input of the functional size approximation process (i.e. functional requirements specifications document of software development projects): these functional requirements specifications are typically documented by the requirements engineers at different levels of granularity. The research issue underlying this thesis and the structure of the thesis are detailed next.

Research issue

Software project managers and technical leaders participate in arranging contractual agreements between the software development organizations and their customers by estimating the effort and duration of the software development projects.

Managers and technical leaders use the available methods to estimate the effort and duration of such software projects with the objective to develop software products in a cost effective and timely manner. However, the use of these estimation methods in an *'a priori'* context faces challenges common to all, such as:

- the lack of detailed and complete software requirements specifications;
- the inability to identify the correct level of granularity of such incomplete and non-detailed software requirements specifications using rigorous criteria; and
- the rapid growth of the size and the complexity of the software projects.

One of the major issues in software effort estimation is the approximation of the functional size of a software product. A few techniques have been proposed to tackle this issue to arrive at an adequate approximation of the software functional size when only high level, incomplete requirements specifications are available.

This size approximation issue normally takes place at the early stages of the software development life cycle and before a significant portion of the product specifications is detailed enough (i.e. it takes place when there is a lack of detailed and complete functional requirements specifications of software development projects) for precise measurement. Therefore, this impairs the software engineer's ability in achieving an accurate measurement of the functional size of the software to be developed: with a lack of precise measurement of this input to effort estimation, it is then challenging to prepare credible effort estimate for the software product to be developed. To address this issue, researchers have recognized the importance of functional size approximation methods in attempts to provide a reasonable approximation of the software functional size at early phases of the software development life cycle.

Obtaining software effort estimation based on measuring the software functionality was first proposed by (Albrecht, 1979). Several methods refining Albrecht's concepts and rules have been standardized by ISO (ISO19761, 2011), (IFPUG20926, 2009).

While the functional size of software development projects can be measured accurately with these ISO standards when all the details of the functionality are available, it is much more challenging and imprecise when the initial requirements are at a high level and lack details.

Some representations and notations of functional requirements specifications like use-case textual specifications and UML diagrams can be used to document the requirements of the users by specifying the functionality that has to be delivered. On the other hand, several studies (Trudel, 2012), (Marín et al., 2010) report quality issues in the documented requirements specifications, in particular at the early phases of the software development life cycle, including for instance:

- requirements incompleteness;
- requirements ambiguity;
- requirements inconsistency; and
- requirements incorrectness.

The lack of detailed and complete specifications of software functional requirements is caused by many factors (Tsumaki et Tamai, 2006), including for instance:

- the insufficient experience and skills of the personnel responsible for elicitation, analysis, specification, and verification of such requirements;
- the ambiguous desires of the stakeholders; and
- in some cases, the unavailability of historical data on similar software projects.

The British Computer Society has reported that 87.3% of 1027 software projects investigated have not been accomplished successfully for many reasons such as incorrect requirements definition, incomplete requirements specifications, requirements inconsistency and lack of knowledge and experience of software project managers (British-Computer-Society, 2000).

During an early phase of the development life cycle of software projects, and more specifically during the approximation process of software functional size, the lack of detailed and complete functional requirements specifications is common, which leads to many challenges, for instance:

- the level of granularity (i.e. the level of details) of the functional requirements specifications of software is identified subjectively using intuition, experience and/or opinions of the field experts;
- scaling factors are not assigned, or assigned subjectively to identify the level of granularity of the functional requirements specifications of software and without following a systematic methodology that is defined for this purpose (Desharnais, Kocaturk et Abran, 2011); and
- there is no standardized notation to define a standard set of scaling factors to be assigned by the requirements engineers to the functional requirements specifications of software development projects to identify their level of granularity.

Thesis organization

This thesis contains eleven chapters (including the introduction and the conclusion). This section presents an outline of the structure of the thesis.

Chapter 1 presents an analysis of the literature related to early approximation of software functional size of software development projects: this includes discussing the literature main contributions, identifying the challenges encountered, and the shortcomings.

Chapter 2 presents an analysis of the several research practices in the literature related to functional requirements analysis and specification: this includes discussing the research practices aimed to support the specification of the functional requirements of software in the early phases of the development life cycle of software projects.

Chapter 3 presents the definition of the research project motivation, the research goal and objectives, as well as the inputs and the users of the research results. This chapter also presents the detailed methodology designed to achieve the identified objectives of this research project.

Chapter 4 presents the conduct of an experimental study to evaluate the reproducibility and accuracy of the approximation results with the latest variant of the *Early & Quick* techniques, the *Early & Quick* COSMIC technique: this will be achieved by asking a group of practitioners with significant experience in the software engineering industry to apply the *E&Q* COSMIC technique to a set of early requirements specifications of a real-time software system.

Chapter 5 presents the identification of the basic building blocks in the proposed framework for scaling factors: this includes identifying the relevant concepts needed for the full-functional specification of the functional requirements of software and defining the format and the type of the scaling factors. This chapter also consists in designing a meta-model to capture the identified relevant concepts with the defined scaling factors. This research phase consists also in designing criteria to identify the level of granularity of the functional requirements of software development projects.

Chapter 6 presents the verification of the usability of the scaling factors framework with groups of practitioners in the software engineering industry. It involves conducting an experimental study with three (3) groups of industry practitioners to apply the scaling factors framework to a single case study.

Chapter 7 presents the definition of the interval scaling factors to be assigned by the scaling factors framework to rank the level of granularity of functional requirements specifications of software in measurements units of the international standard for software functional size measurement: COSMIC - ISO19761. This chapter presents the use of measurement data reported in the literature on the size of functional processes as the basis for setting up the interval scaling intervals with quantitative values.

Chapter 8 presents the verification of the applicability of the scaling factors framework to a variety of case studies by the principal researcher. It involves conducting an experimental study with four (4) case studies that represent software applications that are different in software type, business objective, context, and software functional size.

Chapter 9 presents the approximation of the functional size of four (4) case studies using statistical table of the *E&Q* COSMIC technique after identifying the levels of granularity of the case studies using the scaling factors framework to illustrate the value added of using the interval scaling factors of the framework over using statistical table of the *E&Q* COSMIC technique in approximate sizing.

The conclusion chapter presents a summary of the results of this research project, as well as its contributions and suggestions for future work.

CHAPTER 1

SOFTWARE FUNCTIONAL SIZE IN THE LITERATURE

1.1 Introduction

Cost overruns of software development projects are frequent: software project managers require reliable effort estimates in order to conduct software development projects with minimal risks. Today's software engineering industry works hard to conduct software development projects within a reasonable timeframe and budget. It is considered more risky to fund the software industry than funding railways and car factories whereas software development projects starts with some explicit requirements and many more implicit requirements (Fehlmann et Kranich, 2012).

Software functional size is one of the main input variables for software effort estimation models and techniques: it can be calculated at the early phases of the software development life cycle rather technical measures like the number of source lines of code which can only be applied once the software has been built.

Over the past thirty years, software project managers have used detailed measurement of the functionality of completed software projects in order to build software effort estimation models: with such estimation models, it is feasible to estimate the development effort when the software functional size can be measured by a standardized measurement method like ISO19761 – COSMIC (ISO19761, 2011), when the level of granularity of the functional requirements specifications allows the software measurer to identify the Base Functional Components – BFC, and apply the detailed measurement concepts and rules. However, this is rather late in the development life cycle of software projects and a significant portion of the project budget may already have been spent by the project teams. (Gencel et Demirors, 2008) report that one quarter of the measurement effort is devoted to identify functional transactions (i.e. the functional decomposition of software functional requirements and identification of Base Functional Components).

This chapter is organized as follows:

- section 1.2 presents an overview to software effort estimation;
- section 1.3 presents a detailed analysis of the literature related to the approximation of software functional size and its importance in building effort estimation models at early phases of the development life cycle; and
- section 1.4 presents a summary of the chapter.

1.2 Software Effort Estimation

Research on software effort estimation began with a study in 1965 (Nelson, 1967) for 104 attributes of 169 software projects: this study opened the door to propose effort estimation models and techniques such as: COCOMO (Boehm, 1981), SEER-SEM (Jensen, 1983), PRICE-S (Park, 1988), SLIM (Putnam et Myers, 1992), CHECKPOINT (Jones, 1997), Bayesian approach (Chulani, 1998) and COCOMO II (Boehm et al., 2000). However, all these methods and techniques have faced similar problems, for instance:

- the lack of detailed and complete software requirements specifications; and
- the rapid growing of the size and the complexity of the software projects; making it difficult to estimate the development effort of the software projects.

Software effort estimation models and techniques are used by different stakeholders of a software development project and for different purposes including:

- budgeting: the primary purpose of using these models and techniques;
- trade-off and risks analysis: to eliminate or reduce the risks at different phases of the software development life cycle; and
- software project planning and control: provide the ability for detailed planning of the software project on each activity and milestone.

1.3 Software Functional Size Approximation

A review of functional size approximation techniques has been conducted by (Meli et Santillo, 1999) and (Santillo, 2012): they classified approximation techniques into two (2) main categories: direct and derived functional size approximation techniques:

- the direct approximation techniques are based on analogical reasoning and intuition: they provide functional size approximation results using the experience of the approximator to perform analogy with similar pieces of software and without adopting a step-by-step algorithmic process. Therefore, their approximation results are difficult to justify;
- the derived approximation techniques are well defined algorithmic or structured techniques and are based on theoretical or statistical models. However, they are not suitable for atypical software projects and can only be calibrated using historical data of completed software projects.

1.3.1 Direct approximation techniques

The direct functional size approximation techniques adopt the "Expert Opinion" approach: they completely depend on the expertise of the personnel responsible for the functional size approximation of software. These approximations are influenced by many subjective factors like personal relationships, contractual aspects that are mostly common in teams of collaborating experts. Therefore, it may result in reasonable functional size approximations but it is challenging to recognize when it is reasonable, and when it is not. The following shows some examples of these functional size approximation techniques:

- delphi technique (Brown, 1968): it considers a group approximation approach rather than approximation conducted through individual approximations. For example, each individual involved in the approximation technique constructs anonymous approximations, and next these individual approximations are combined to achieve the overall size approximation as a group estimate;
- three-point approximation technique (Keefer et Bodily, 1983): it considers improving the direct approximation by using more information from the personnel involved in the

functional size approximation. Given the minimum, most likely, and the maximum values for the functional size, the approximated size value can be calculated using the following equation: approximated-size = (min + 4 × most-likely + max) / 6.

1.3.2 Derived approximation techniques

A few derived functional size approximation techniques with an algorithmic model have been proposed, including:

- extrapolative approximation technique (Tichenor, 1998): it is applied by asking each person involved in the functional size approximation to approximate one functional component and derive the remaining approximations through a statistical or theoretical basis;
- sampled approximation technique: it derives the IFPUG (IFPUG20926, 2009) approximation for one portion of the system, and uses this approximation to extrapolate the remaining portions of the system;
- average complexity approximation technique (Jones, 1986): it identifies functional components (such as: External Input (EI), External Output (EO), External Inquiry (EQ), Internal Logical File (ILF), or External Interface File (EIF)) to approximate the functional size according to these identified components. For example: using the ISBSG Benchmark (ISBSG, 2003) average size of functional components, the function size is approximated using the following formula: approximated-size = 4.3 × #EI + 5.4 × #EO + 3.8 × #EQ + 7.4 × #ILF + 5.5 × #EIF – where #EI represents the number of External Inputs; and
- *Early & Quick* COSMIC Technique: it was initially designed by (Meli, 1997) for the Function Points Analysis (FPA) method. Then, the *Early & Quick* COSMIC technique was proposed in 2000 (Meli, 2000) based on the initial design of the technique. After that, the *Early & Quick* COSMIC was generalized in 2004 (Conte, Iorio et Santillo, 2004): further details of this technique are presented in Chapter 4.

Generally, functional size approximation techniques can be applied using the bottom-up or the top-down software decomposition approaches to the software systems under study:

- the bottom-up approach starts by approximating the lowest level components to achieve low-level size approximations; these low-level approximations are combined into higher level size approximations; and
- the top-down approach starts with the overall software to achieve an approximation for the total software size; the lower levels components' are next approximated as a relative portion of the full size approximation.

1.3.3 Studies on functional size approximation

Several studies have proposed to approximate software functional size at an early phase of the life cycle of software projects by proposing mapping between requirements specifications documented using a requirements modeling language such as the Unified Modeling Language (UML) (Booch, Rumbaugh et Jacobsen, 1999):

(Bévo, Lévesque et Abran, 1999) proposed mapping between concepts of a subset of UML diagrams (i.e. use cases, scenarios, and classes) and concepts of the COSMIC measurement method:

- boundary of the system is included in the use case diagram boundary;
- a use case corresponds to a functional process;
- data movements are represented in scenarios: interactions that occur within a use-case;
- a class of the class diagram corresponds to a data group and the attributes of those classes correspond to the data attributes; and
- an actor corresponds to a functional user.

However, triggering events and layers are not represented with concepts in UML diagrams.

(Jenner, 2001) discussed the concept of granularity in use-cases of the (Bévo, Lévesque et Abran, 1999) proposal:

- a functional process is represented by sequence diagram; and
- data movements are represented by the interaction messages of the sequence diagram.

(Poels, 2003a) proposed a mapping between the concepts of the COSMIC measurement method and the business and service models of MERODE (Dedene et Snoeck, 1994). After that, this proposal was extended to allow the measurement of multi-layer software applications (Poels, 2003b):

- the users of business model correspond to services model;
- the boundary of the business model corresponds to the boundary between business model and the users;
- the functional process of business model corresponds to a set of class methods over all of the enterprise objects;
- a data movement corresponds to each class method that composes a functional process;
- data groups correspond to the classes of business model;
- the users of the services model correspond to user interface model;
- the boundary of services model corresponds to boundary between services model and the users; and
- a functional process of the services model corresponds to a non-persistent service object.

(Nagano et Ajisaka, 2003) proposed a procedure to measure the functional size of real-time software applications specified using xUML (Mellor, Balcer et Jacoboson, 2002):

- parameters of messages and control signals are candidate data groups;
- triggering events are identified in the collaboration diagrams: include relationship between the external entity and the objects of the system;
- functional processes correspond to a sequence of data movements; and
- data movements correspond to the actions that an object performs to move it from one state to the next state in a collaboration diagram.

(Azzouz et Abran, 2004) proposed a tool to automate the measurement of the functional size of software applications documented using the Rational Unified Process (Kruchten, 2000) in an extension to Bévo's and Jenner's proposals:

- a layer cannot be represented in the UML diagrams: the user of the tool must manually identify the layers of the system; and

- a new stereotype is needed to identify the triggering events in the use-case diagrams.

The proof-of-concept prototype tool was designed in accordance to the direct mapping between COSMIC measurement method (ISO19761, 2011) and the UML (Booch, Rumbaugh et Jacobsen, 1999) concepts and notation as shown in figure 1.1. This allowed the Rational Rose artefacts to be directly extracted and used in the functional size approximation procedure.

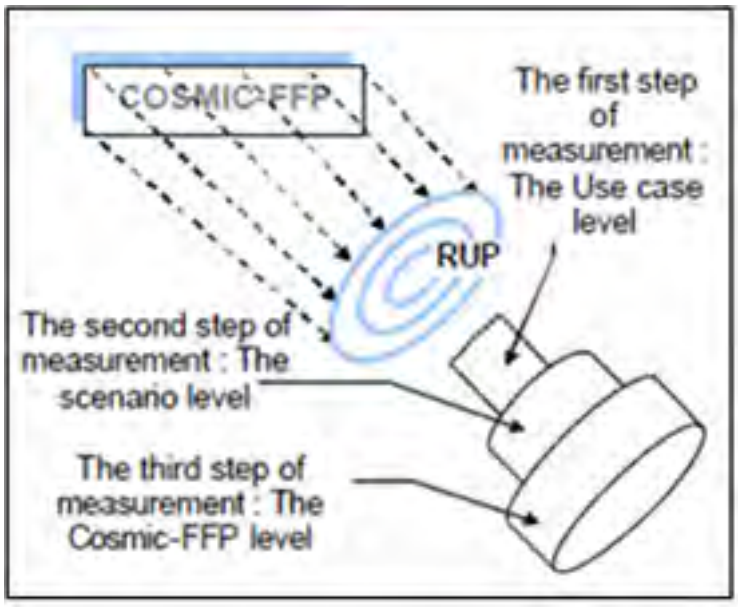

Figure 1.1 COSMIC and the RUP mapping
source: (Azzouz et Abran, 2004)

Adopting this design methodology resulted in a proof-of-concept prototype capable to derive early approximations of software functional size when only high-level information is available. Figure 1.2 depicts the tool architecture suggested by (Azzouz et Abran, 2004):

- the left-hand side represents the concepts, procedures, and rules required to automate the approximation of the functional size; and
- the right-hand side represents the environments that will provide the approximation inputs.

As mentioned earlier, the artifacts are extracted from the Rational Rose tool environment: these artifacts correspond to different levels of details in the requirements space. This multi-level phenomenon has been approached through adopting multi-level approximation as illustrated in table 1.1. It is clear that each level has its own level of granularity and its own unit of approximation. The first two levels provide the early approximation of the system's functional size and the third level provides more precise and accurate approximation results using the international measurement conventions of the COSMIC measurement method.

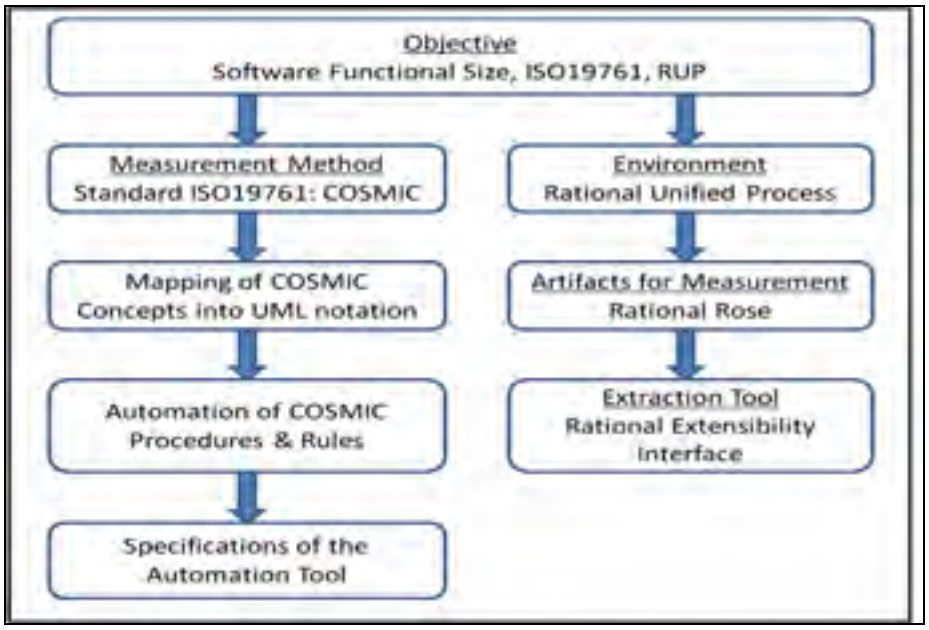

Figure 1.2 Architecture of a software approximation automation tool
source: (Azzouz et Abran, 2004)

Table 1.1 Distinct size units for different levels of granularity
source: (Azzouz et Abran, 2004)

Development Phase	**RUP Artifacts**	**Unit Convention**
Business Modeling/ Requirements Analysis	Use-case Diagrams	
Analysis/Design	Scenarios	
Analysis/Design	Detailed Scenarios	

At the business modeling & requirements analysis level use-case diagrams are the artifacts used to calculate a level of functional size. Hence, the size unit at this level is labeled - see table 1.1: Use case functional size unit (Ufsu).

At the analysis level the scenario diagrams are the artifacts used to approximate another level of functional size. Hence, the size unit at this level is labeled - see table 1.1: Scenario functional size unit (Sfsu).

Finally, the analysis/design levels the detailed scenario diagrams are the used artifacts to approximate the functional size at the level of granularity prescribed by the ISO 19761 international standard. Hence, the size unit at this level corresponds exactly to the COSMIC functional size unit (Cfsu) as adopted in the 2003 version of the ISO19761 standard. It is worth mentioning that since 2011 the size unit Cfsu has been modified to COSMIC Function Point (CFP).

While this approach has recognized different levels of granularity and corresponding different size units (Ufsu, Sfsu and Cfsu) this research project did not tackle the lack of available data and the convertibility – or scalability – across these 3 distinct size units.

(Diab et al., 2005) proposed a set of formal rules to allow the measurement of real-time applications that are documented using the real-time object-oriented modeling (ROOM):

- a boundary of the system is represented by a set of actors;
- layers correspond to a set of actors with the same level of abstraction;
- a transition corresponds to a functional process;
- data movements are represented by actions and messages;
- actors and protocol classes correspond to data groups; and
- attributes and variables of classes correspond to data attributes.

(Habela et al., 2005) proposed an extension to the use-case model to allow the measurement of the functional size using the COSMIC measurement method:

- a use case corresponds to one or more functional processes;
- data movements are identified in each step described in the scenarios; and
- use, extend, and generalization relationships between use-cases are taken into account to avoid redundancies in the measurement.

(Condori-Fernández, Abrahão et Pastor, 2007) proposed a procedure to estimate the functional size of object-oriented systems from requirements specifications documented using the OO-Method approach:

- the boundary of the system corresponds to the border between the set of use-cases and the actors of the use-case diagram;
- a functional process corresponds to each elementary function of the functions refinement tree;
- data groups are identified in the sequence diagram;
- an actor, control class or entity class of the sequence diagram corresponds to a data group;
- data movements correspond to the messages of the sequence diagram; and
- a single layer is identified and triggering events are not represented.

(Grau et Franch, 2007a) proposed a set of mapping rules to measure the functional size of i* models generated using PRiM (Grau et Franch, 2007b):

- a boundary corresponds to the actor of the operational i* model;
- users are actors of the operational i* model;
- data movements are identified in the operational i* model and correspond to any dependency where the *dependum* is a resource;
- a functional process corresponds to an activity of the detailed interaction scripts;
- triggering events are part of the conditions associated to the activity; and
- data groups correspond to the resources of the detailed interaction script.

(Levesque, Bevo et Cao, 2008) proposed to apply the COSMIC measurement method to measure the functional size of software from use-case and sequence diagrams: their proposal classifies the functional processes into data movement types and data manipulation types:

- use case is a functional process;
- actors of the use case are the users;
- entities of the sequence diagram are data groups;
- data movements correspond to the messages among the entities of sequence diagram; and
- data manipulations are the conditions associated to error messages of sequence diagrams.

(Marín et al., 2008) proposed a procedure to measure the functional size of object-oriented systems generated in Model Driven Architecture (MDA) environments from their conceptual models structured in the three phases of the COSMIC method:

- the scope of measurement is determined for functional processes, layers, or whole application;
- the layers correspond to the hierarchical tiers of the OO-Method applications;
- the pieces of software correspond to three (3) components: client component, server component, and database component;
- the users are the human users, client component, and server component of applications and they are separated by a boundary;
- the functional processes are groups of functionality that can be accessed by the user and corresponds to the interaction units;
- data groups correspond to classes of the object model; and
- data attributes correspond to the attributes of the classes.

The authors of this measurement procedure proposed sixty-nine (69) rules to identify the data movements that can occur in the OO-Method applications and finally the measurement procedure proposed a set of rules to obtain the functional size of each functional process, each piece of software, and of the whole application.

(Gencel, 2008) explored the benefits of defining and using a vector of measures by identifying elements (i.e. the BFC types of the COSMIC measurement method: Entry; Read; Write; and eXit) of functional size instead of a single size value on the estimation of productivity and effort values through conducting an experimental study with three (3)

software applications in which their functional size range from 164 CFP to 229 CFP. The results showed that building estimation models using a vector of measures for functional size rather than on a single value is promising. However, more software projects are needed to be used in experimentation to confirm or disconfirm this hypothesis from their study.

(Abrahao et Insfran, 2008) proposed a measurement procedure to automatically approximate the functional size of object-oriented requirements specifications: it proposed a set of measurement rules to map the concepts of the requirements meta-model and the Function Point Analysis meta-model. However, no experimental studies were conducted with participants to verify the usability or the applicability of the proposed procedure. Furthermore, the proposed meta-model does not take into consideration the identification of different levels of granularity of software requirements specifications.

(Kassab, Daneva et Ormandjieva, 2008) proposed a meta-model aimed to complement the functional requirements dimension with the non-functional requirements to be used in effort estimation techniques: the proposed meta-model is aimed to extend the use of the COSMIC functional size measurement method to measure the size of the non-functional requirements to result in effort models as a function of size of functional and non-functional requirements. On the other hand, the proposed meta-model does not take into account that in practice, software requirements are modeled at different levels of granularity and it assumes that it is composed of only one (1) function/operation.

(Lavazza et Bianco, 2008) applied the Function Point Analysis (FPA) method to user requirements represented using problem frames in native notation and using UML to measure their functional size. However, the presented approach relies on the concepts of FPA to measure the functional size of problem frames (narrative and UML) notation: the measurement process requires that the measurer explore the documentation to find the base functional components (BFC) which means that such a process is still a subjective one. The approach neglects the fact that software requirements are normally documented at different levels of granularity.

(Lind et Heldal, 2009) conducted a research study aimed to present the usage of UML component diagrams to approximate software code size using the COSMIC measurement method. These authors have designed a linear model to approximate the code size of software components within seven (7%) percent accuracy: it is worth mentioning that this linear model is valid for software components of size between four (4) and twenty-six (26) COSMIC Function Points.

(Sellami et Ben-Abdallah, 2009) proposed an approach for fine-grain measurement of requirements documented using the UML use-case diagram and their associated scenarios: these authors indicated that a detailed measurement of the use-case diagram gives a reference measurement to other UML diagrams and a use-case shall contain three (3) types of scenarios as follows:

- nominal scenario: it realizes the post-conditions of use-case in a natural and frequent way;
- alternative scenario: it meets the post-conditions of use-case but via a redirected or rare way and;
- error scenario: it does not realize the post-conditions of use case.

The authors map each use-case diagram into a corresponding directed graph (G) which can be decomposed into sub-graphs that represent the three (3) types of scenarios and instantiated by an actor (A) that interacts with a use-case (U) or by another use-case that has:

- include relationship: inclusion arc (“Include”) is followed in its direction;
- extend relationship: extension arc (“Extend”) is followed in its opposite direction; and
- inheritance relationship: inheritance arc is followed in both directions.

The functional size of a use-case diagram (G) is equal to the sum of the sizes of its sub-graphs and the functional size of a use-case U: the three types of relations (inclusion, extension and inheritance) that U has with other use-cases U’ in the same sub-graph. The functional size of a triggering event is equal to (1) CFP. Finally, the functional size of a scenario in a use-case is the total number of messages exchanged between internal and external objects of the scenario sub-graph.

(Condori-Fernandez et al., 2010) explored the predictability of software project size from software product size by taking into account the size of the functional and non-functional requirements by conducting an experimental study with fifty-five (55) undergraduate students divided into eleven (11) groups to build estimation models of web-application development projects by:

- measuring the functional size of the functional and non-functional requirements using the COSMIC measurement method and its extension in (Kassab et al., 2007); and
- calculating project size using the Project Size Unit model.

The results of this experimental study showed that there is a causal relationship between the software project size and software product size. However, the authors suggested conducting more experimental studies to obtain more results to confirm or disconfirm their hypothesis.

(Hussain, Kosseim et Ormandjieva, 2010) proposed an approach aimed to approximate the functional size of use-cases that are written in a format similar to agile processes format and without the need to formalize these textual requirements: the proposed approach requires the use of historical organizational datasets which contain for each textual requirement specification a corresponding functional process measured using the COSMIC measurement method. However, the proposed approach has not been tested using textual requirements documented at different levels of granularity which raise concerns with respect to the generic validity of the proposed approach in this context.

(Lind et al., 2011) proposed an automated tool meant to approximate code size based on a previously defined UML profile aimed to collect all the necessary information from UML component diagrams to allow applying the COSMIC functional size measurement method. The proposed automated tool is designed to allow approximating code size from UML diagrams other than component diagrams. An experimental study was conducted to evaluate the automated tool in approximating code size from requirements specifications and software implementations. This paper claims that the automated tool has parsers that allow the approximation of code size from use-cases and sequence diagrams but no case studies at the

date of publication are presented to support this claim. Further, the tool has not been applied at the early phase of software development life cycle: only high-level and incomplete requirements specifications are available which is different than UML component diagrams that appear relatively later in the software development life cycle.

(Top, Demirors et Ozkan, 2009) explored the types of measurement errors committed during the measurement process of software application using the COSMIC measurement method: it aimed to identify the reasons and to illustrate their impact on the accuracy of the functional size measurement results. The authors conducted an experimental study of measuring twelve (12) industrial software applications by five (5) different inexperienced measurers who had six (6) hours of training in using the COSMIC measurement method: the authors suggested that knowledge and level of experience of the measurers involved are primary impact factors on the quality of the measurement results.

(Heeringen, Gorp et Prins, 2009) presented an overview of the differences between functional size approximation methods in terms of accuracy of the results and time required to conduct such measurement:

- the results of functional size approximation reported that when using the COSMIC "Equal size bands" approach there is an average size difference of 1.26% from results reported using the detailed COSMIC method with a standard deviation of 10.31 and 90 percent of the functional size approximation results using the COSMIC "Equal size bands" approach fall within a range of -15% to +25% of the results calculated using the COSMIC measurement method; and
- the results reported in this paper also found that using the COSMIC "Average functional process" approach, there is an average difference of (1.98%) from results reported using the detailed COSMIC measurement method with a standard deviation of 36.36 and 55 percent of the functional size approximation results using the COSMIC "Average functional process" approach fall within a range of (-25%) to (+50%) of the results calculated using the COSMIC measurement method.

(Desharnais, Kocaturk et Abran, 2011) used the COSMIC measurement method to evaluate the quality of functional requirements specifications. The authors (Desharnais, Kocaturk et Abran, 2011) proposed criteria that are based on the documentation coverage for each artifact of the functional user requirements: these criteria take advantage of the information available during the measurement of the functional user requirements to classify each identified Functional Process on an ordinal scale in decreasing order (A → E) as follows:

A. the functional processes are completely documented including the data model used throughout the set of functional processes;
B. the functional processes are documented but without a precise data model making it challenging to identify all data movements of distinct data groups;
C. the functional processes are identified but without details to allow the identification of the precise number of individual data movements;
D. the number of functional processes is explicitly stated; and
E. the functional process is implicit in the documentation.

Table 1.2 presents the results of the functional size approximation of eight (8) use case scenarios from (Desharnais, Kocaturk et Abran, 2011). The functional size approximation is conducted on three (3) versions of use case scenarios where the level of granularity in the 3rd version (i.e., version 3.0) is more detailed than the level of granularity in 2nd version (i.e., version 2.0) and the level of granularity in the second version is more detailed than the level of granularity in the 1st version (i.e., version 1.0).

The results presented in table 1.2 shows an average increase of 37% when the functional size of the scenarios in version 2.0 is compared to the functional size of the scenarios in version 1.0 and an average increase of 26% when the functional size of the scenarios in version 3.0 is compared to the functional size of the scenarios in version 2.0: such increase in the functional size approximation leads the principal researcher to the following observations:

- there is no usage or undefined usage of scaling factors during the process of functional size approximation of the three (3) versions of use case scenarios;

- the early identification of the level of granularity of the use case scenarios is a key factor in the process of functional size approximation; and
- there is a need to develop a methodology that identifies the level of granularity of the use case scenarios, and proposes a set of scaling factors to rank the level of granularity of such scenarios.

Table 1.2 Functional size approximation of 3 versions of use case scenarios
source: (Desharnais, Kocaturk et Abran, 2011)

User Scenario Id. Code	Approximated functional size of version 1.0 (in CFP) (1)	Approximated functional size of version2.0 (in CFP) (2)	Percentage of increase in functional size (1) & (2)	Approximated functional size of version 3.0 (in CFP) (3)	Percentage of increase in functional size (2) & (3)
US1	13	28	115%	34	21%
US2	4	8	100%	16	100%
US3	9	9	0%	9	0%
US4	6	8	33%	12	50%
US5	9	8	-11%	8	0%
US6	12	14	17%	16	14%
US7	12	14	17%	16	14%
US8	12	15	25%	16	7%
Total	**77 CFP**	**104 CFP**		**127 CFP**	
Average increase of functional size			**37%**		**26%**

(Soubra et Abran, 2012) emphasized that all COSMIC-based functional size measurement procedures have to produce the same measurement results when they are applied to the same set of functional user requirements. The authors proposed a revised set of rules to refine the functional size measurement procedure in (Soubra et al., 2011): the aim of such refinement is to resolve the variance of the measurement results of real-time software requirements documented using the Simulink modeling tool.

1.4 Chapter Summary

The reliability of the results of the software effort estimation process is a major factor on the success of software development projects: the effort estimates are used for planning and control for the development activities of software projects. Software functional size is one of

the primary inputs for the *'a priori'* software effort estimation models and techniques. There are few techniques to approximate software functional size (COSMIC, 2007). However, all these techniques face a common challenge: the lack of detailed and complete software requirements specifications especially at the early phases of the software development life cycle.

The current practice in the software measurement community is to accept incomplete and non-detailed requirements specifications documents (i.e. to have portions of requirements documented at different levels of granularity) and without systematically handling this diversity by either the requirements engineers or software size approximators. The levels of granularity of functional requirements specifications are subjectively identified using intuition, experience or opinion of the field experts.

CHAPTER 2

SOFTWARE REQUIREMENTS QUALITY IN THE LITERATURE

2.1 Introduction

Producing high-quality software products in a cost-effective way requires control over all the phases of the software development life cycle. One of the most important phases in the development life cycle of a software product is the requirements engineering phase (Abran et al., 2004): this phase includes requirements elicitation, requirements analysis, requirements specification, and requirements validation.

The requirements engineering phase gained its importance during the development life cycle of a software product since many stakeholders (e.g. project manager) use different versions – based on the availability of such versions – of requirements specifications documents for different uses: one of the main uses is to early approximate the functionality of the software product to be developed in order to build effort estimation models to estimate the required effort to develop the software product in a cost-effective manner (i.e. within pre-defined time and cost constraints). Therefore, the quality of the requirements specifications document that results from the requirements engineering phase (Abran et al., 2004) is highly important for the success of software development projects. The quality of the software requirements specifications highly depend on the expertise of the software engineers who elicit and document the software requirements in terms of writing skills, knowledge of the project domain, and the complete understanding of software requirements. Other challenges especially at early phases of the software development life cycle include ambiguous and inconsistent requirements, poor user collaboration, unfixed or fluctuating requirements. All these challenges highly affect the production of well-detailed and complete software requirements specifications document which is an important attribute of a requirements specifications document as recommend in the IEEE-830 standard of recommended practice for software requirements specifications (IEEE, 1998).

This chapter is organized as follows:

- section 2.2 presents the definitions of software requirements and of the requirements elicitation process;
- section 2.3 presents software requirements modeling using the Unified Modeling Language;
- section 2.4 presents the definition of software requirements quality and detailed analysis of the literature related to software requirements quality that aim to maintain or verify the quality of software requirements; and
- section 2.5 presents a summary of the chapter.

2.2 Software Requirements & Elicitation Process

2.2.1 Software requirements

A requirement is defined as a property that must be exhibited in order to solve a real-world problem (Abran et al., 2004). The Functional User Requirements (FUR) is a subset of the user requirements: they represent the functional user practices and procedures that the software must perform in order to fulfill the users' needs (ISO14143-1, 2007). On the other hand, the non-functional requirements (NFR) characterize the software constraints which include quality and technical requirements (e.g. maintainability, portability, security, and privacy…etc).

2.2.2 Software requirements elicitation process

The main objective of the requirements elicitation process is to identify the sources of the software requirements. The elicitation process includes the usage of elicitation techniques by the software engineers to collect these software requirements: this process is typically expressed as "requirements capturing", "requirements discovery", and "requirements acquisition".

2.2.2.1 Software requirements sources

The software requirements have many sources: it is crucial to identify all the potential sources of requirements and specify its impact on the software projects. The SWEBOK guide (Abran et al., 2004) lists the main sources of requirements:

- goals: sometimes called "business concern", "critical success factor"; the software engineers need to assess the values and the costs of such software goals by means of a feasibility study;
- domain knowledge: the software engineers should acquire knowledge about the software domain – such knowledge helps the software engineers to assess the trade-offs between conflicting requirements;
- stakeholders: the software engineers have to identify, express, and manage the software requirements from different types of stakeholders;
- operational environment: the software engineers have to obtain information from the operational environment because it affects the software project feasibility, cost, and design choices; and
- organizational environment: the software engineers should take into consideration that the new software should not lead to unplanned change on the business process.

2.2.2.2 Software requirements elicitation techniques

This activity is normally conducted after the identification of the requirements sources: the following techniques from (Goguen et Linde, 1993) present some of the techniques that can be used to elicit the requirements from project stakeholders:

- interviews: this technique has been widely used to acquire requirements from stakeholders in variety of domains: it includes questionnaires, open-ended interviews and focused groups;
- scenarios: this technique allows the software engineers to assemble a framework of questions about different scenarios about the stakeholders' tasks. The most common scenario notations are the use case scenarios: they provide a connection between the

intended software and its external environment in order to understand the software's context and identify the interfaces with its operational environment;

- prototypes: this technique provides means to validate the software engineers' interpretation of the stakeholders' requirements and to acquire new requirements. However, the prototypes may distract the stakeholders' attention from focusing on the software core functionality to cosmetic and quality issue;
- facilitated meetings: this technique is applied by assembling groups of stakeholders meetings to discuss their software requirements needs. The objective of these groups meetings is to reach a consensus and consistency about the stakeholders needs. The facilitated meetings provide the opportunity for brainstorming and refinement of ideas, which may be difficult to practice using interviews. However, these meetings should be managed carefully in order to prevent groups' loyalty and senior members from dominating these meetings; and
- observation: this technique is relatively expensive, but it provides a way to understand the software context within the organizational environment by observing how users interact with their software and with each other.

On the other hand, the requirements elicitation process is not without its difficulties: one research study by (Tsumaki et Tamai, 2006) reported 22 different sources of difficulties, for instance:

- incomplete requirements;
- incomplete understanding of needs;
- incomplete domain knowledge;
- poor users' collaboration;
- overlooking tacit assumptions;
- incorrect requirements;
- ill-defined system boundaries; and
- ambiguous requirements.

2.3 Software Requirements Modeling

2.3.1 The Unified Modeling Language

The Unified Modeling Language (UML) is an industry defacto standard from the Object Management Group (OMG): it is a software modeling language aimed to visualize, specify, construct, and document the software requirements (Booch, Rumbaugh et Jacobsen, 1999). Therefore, it is not a programming language even though there have been some trials to use it as an executable language. The UML has four (4) primary objectives, as follows:

- visualization: the UML provides the ability to depict the functional requirements of the software system, the solution, and the architecture of such software system. The UML is also capable to visually present the business processes and the software elements through the entire software development life cycle;
- specification: the UML enhances the quality of the modeling by enabling the integration of additional descriptions of the visual models;
- construction: the UML provides the ability for software construction through generating code from the visual models; and
- documentation: the UML provides the ability to enhance the specification and the visualization of the visual models through additional documentation of the software artifacts.

2.3.2 Use case scenarios

Use-case scenarios are prepared by the software engineers to document the functionality (i.e., the software functional requirements) that has to be delivered by a software product: this is done at an early phase of the software development life cycle. The use-case scenarios describe the internal behavior of the use-cases identified in use case-diagrams: they are intended to provide a list of all the sequential interactions between the users of the software and the software product.

The use-case scenarios should have an identifiable beginning and end of the scenario, and describe all possible variants of interaction, such as: the main success scenario, alternative scenarios, and error sequences. The OMG define a use-case scenario as follows: *"A scenario represents a particular succession of sequences which is run from beginning to end of the use case. A scenario may be used to illustrate an interaction or the execution of a use case instance"* (Booch, Rumbaugh et Jacobsen, 1999).

The literature refers to many templates to provide a description of the use-case scenarios: the following template (Roques, 2004) presents a sample one that describes the major parts of a use-case scenario:

- identification summary: it includes title, summary, creation and modification dates, version, personnel in charge, and actors of the use case scenario;
- flow of events: it describes the main success scenario, the alternative and error sequences, as well as the pre-conditions and the post-conditions;
- user interface requirements: it describes the graphical user interface constraints and depict screen copies; and
- non-functional constraints: it describes the non-functional constraints related to the use case scenario such as availability, accuracy, integrity, confidentiality…etc.

2.4 Software Requirements Quality

There are a number of proposals in the literature to evaluate the quality of the software requirements specifications documents. The IEEE computer society has published a standard (IEEE, 1998) for software requirements specifications called: *"IEEE Recommended Practice for Software Requirements Specifications"*: this standard presents recommended approaches for the specification of software requirements to produce unambiguous and complete specification document. It also includes the required attributes that a software requirements document should have in order to help software customer to accurately describe what they wish to obtain and help software suppliers to understand exactly what the customer wants.

However, this IEEE-830 standard (IEEE, 1998) has not been updated since 1998 to take into account subsequent research initiatives on this topic such as those presented next.

(Kamata et Tamai, 2007) investigated thirty-two (32) software development projects started and completed during the period of 2003-2005 in a large business application software development division of a company in Tokyo to identify the relationship between requirements quality and project success or failure. The quality of the software requirements specifications and overall project performance in terms of cost and time overrun are evaluated by software quality assurance teams and led to the following observations:

- there is a relationship between SRS quality and project outcomes;
- descriptions of SRS in successful projects tend to be balanced; and
- early identification of requirements can be a good way for preventing cost overrun.

(Kujala et al., 2005) reported results of interviews and survey conducted to investigate the role of user involvement in defining user requirements in software development projects: the survey involved eighteen (18) software practitioners working in software-related development projects in thirteen (13) companies and interviewed eight (8) software practitioners working in three different companies. The analysis of the interviews and the survey data showed that early user involvement is related to better requirements quality and showed that involving users and customers as the source of information is related to project success.

(Bjarnason, Wnuk et Regnell, 2011) reported an interview conducted with nine (9) practitioners in a large software development company and questionnaire with a different set of seven (7) practitioners who assured that communication between the stakeholders of a software project is a challenging issue and, most of the time, results in miss-interpretation of software functional requirements and leads in failure to meet the user's expectations. The interviewed practitioners have reported the following reasons for communication gaps:

- scale: increases the challenge of requirements communication;
- common views: weak understanding of other's roles and responsibilities cause gaps;

- temporal aspects: lack of continuity in requirements awareness during the project; and
- decision structures: weak or unclear visions for the software development.

(Dzung et Ohnishi, 2009) presented an ontology for requirements elicitation of a problem domain: this ontology includes functional hierarchy and relationship between functions. Using the ontology (Dzung et Ohnishi, 2009) proposed a method of checking the quality of an SRS document especially the correctness and the completeness attributes. The requirements ontology represents:

- functional hierarchy of a certain software system;
- relationships among functional requirements; and
- attributes of functional requirements.

Each Functional requirement that has one verb and nouns becomes a node of the requirements ontology and relationships including inheritance and aggregation can be represented in the functional structure of a system belonging to a certain problem domain. A requirements ontology includes relationships: mutual complementary, inconsistency between functional requirements and attributes of functional requirements such as agent of the function (who), location of the function (where), time (when), reason (why) and non-functional requirements.

Three steps to check requirements using the proposed requirements ontology are as follows:

- parse initial requirements;
- map requirements to object on ontology; and
- check the requirement using relationships on ontology and using rules.

On the other hand, (Dzung et Ohnishi, 2009) presented no justification on how such meta-model is constructed (also the checking process of requirements using rules): the parsing tools are used with no independent checking of their performance and parsing quality. The approach is checked using only one case study and there is no proof of usability or

applicability of this approach and there is no explanation related to the identification of the levels of granularity of the software requirements.

(Wu et al., 2009) proposed a hierarchical organization model aimed to illicit complete and consistent user's requirements from business models: the upper part presents a layer by layer top-down decomposition of the organization into administrative units and the lower part presents a work-flow model for business processes. During the decomposition process, only units that use software are included in the model and other units are not allowed to enter the model.

Only business processes that run with software are allowed to enter the work-flow: each business process of the work-flow splits into two rows: the upper row depicts work-flow structure of the process in terms of a directed graph of activities and the lower row is filled by information processing tasks corresponding to activities and every activity should link to the user who is obliged to carry on this activity. The aim is to allow only business processes that run with software to keep in the model all the elements and relations relevant to specifying the user's requirements while all the irrelevant ones are excluded and therefore help the software engineer to identify all users of the software and illicit their requirements completely and consistently.

2.4.1 Studies for high-quality software requirements

There have been in the literature some trials to define the quality of the conceptual models and each has used different terminologies to refer to the same concept. (Moody, 2005) proposed a definition of quality that is based on the ISO 9000 standard definition of product or a service: "The total of features and characteristics of a conceptual model that bear on its ability to satisfy stated or implied needs".

(Kececi et Abran, 2001) proposed an approach to identify the correctness of the software requirements specifications by using a logic-based dynamic framework. The approach

adopted a structured procedure to arrange the software requirements specifications into a graphical framework. Therefore, it provided a means of evaluating the requirements specifications clarity, presence, or absence. Further, the procedure allowed tracing (forwards and backwards) of the specific entities from user requirements to design and also provided mapping into a multi-level detailed system and software functionality from inputs and outputs.

(Engels et al., 2002) proposed dynamic meta-modeling rules as a candidate notation for the consistency of requirements specifications and used the dynamic meta-modeling rules to propose the necessary concepts for an automated testing environment. The consistency notation of UML dynamic diagrams was addressed as: "Whenever a model employs different dynamic diagrams, it has also to be determined if and how the different aspects represented in them fit together to express the behavior of the objects involved".

(Santander et Castro, 2002) proposed guidelines to support the integration of i* organizational models and use-case modeling: it describes heuristics to assist requirement engineers to develop use-cases based on i* organizational models by offering two (2) models to represent organizational requirements:

- strategic dependency model: it focuses on the intentional relationships among organizational actors; it consists of a set of nodes and links connecting them where nodes represent actors and each link represents the dependency between actors; and
- strategic rationale model: it is used to understand how systems are embedded in organizational actor's routines.

The proposed approach is to derive use-cases from the i* organizational models in three (3) steps which represent the discovery of system actors and its associated use-case diagrams and descriptions: in steps 1 and 2, the input is the strategic dependence (SD) model and the description of scenarios for use-cases (step 3) is derived from elements represented in the strategic rationale (SR) model. The result of the integration process is use-case diagrams for the intended system and scenario textual descriptions for each use-case.

The proposed approach (Santander et Castro, 2002) generates use-cases and their scenarios from the i* organizational models and suggests guidelines for this process. However, no information was provided to the readers how those i* organizational models are generated (automatically or by software engineer) and at what level of granularity i* organizational models are generated (i.e. what are the quality guarantees that i* models provide to the software engineers).

(Correa et Werner, 2004) proposed a scenario-driven approach aimed to produce detailed use-case specifications to define all transactions in a given use-case as follows:

- definition of the main results expected for each transaction in a use case;
- formalization of all computation and inference rules behind the production of those results;
- exploration of all rules that could deny or restrict the transaction execution; and
- detailing all interactions between the actors and the system that will trigger the use-case transactions and considers non-functional aspects and their influence on the use-case interactions.

The approach proposed by (Correa et Werner, 2004) is applied with only one (1) industry project in the financial domain: it depends on the information available on use-case specification of a use-case and recommends to apply it at an early phase of the SDLC when not all the information are available.

(Buhne et al., 2004) introduced the concepts of goals, scenarios and requirements at four (4) different levels of granularity for the specification of requirements in the context of a development project at DaimlerChrysler as follows:

- vehicle level: defines the requirements for the vehicle with their interrelations;
- system level: defines the requirements for one system consisting of different applications;
- function level: defines the requirements for one application and its functionality; and
- software level: defines the requirements for the implementation of functions.

On the other hand, this approach of (Buhne et al., 2004) does not present a methodology to be followed by the engineers at DaimlerChrysler on how to decompose project requirements on each level of the four (4) levels of granularity.

(El-Attar et Miller, 2006) proposed a process called AGADUC to generate activity diagrams that represent the embedded workflows in use-case textual descriptions aimed to provide information regarding how use-cases are dependent on each other by graphically depicting these workflows within the use-case diagram. However, the proposed process by (El-Attar et Miller, 2006) employs use-cases without checking on their quality: their proposal is verified only with a small case study of three use-cases and no real case-studies from the industry were used to check the proposal applicability to different case studies.

(Pauli et Xu, 2006) conducted a study aimed to integrate the security requirements together with functional requirements: (Pauli et Xu, 2006) identified three (3) types of cases in software system: use-case, misuse-case, and mitigation use-case:

- use-case: case of list of actions taken by a system upon receiving a request from its users;
- misuse-case: the inverse of a use-case which describes the process of executing a malicious act against a software system; and
- mitigation use-case: use-case mitigates the chance that a misuse case completes successfully.

(Pauli et Xu, 2006) considered the decomposition of these use-cases by decomposing one use-case type at a time and then to integrate the decomposed use-cases together to ensure each case type is accurately decomposed before integrating with other case types: the decomposition process is driven by the textual descriptions of each use-case including the identification “includes” and “extends” relationships as follows:

- identify candidate cases from textual descriptions;
- create initial textual descriptions for each;
- identify and model “includes” relationship;
- identify and model “extends” relationship; and

- identify and model appropriate actor assignments.

Then, integrate different case types together after decomposition has occurred for all three case types:

- integrate misuse and use-cases together by using the “threatens” relationship where a use-case will be “threatened” by a misuse case; and
- integrate mitigation and misuse-cases together by using the “mitigates” relationship where a misuse-case will be “mitigated” by a mitigation use-case.

On the other hand, the decomposition and the integrations processes depend on the expertise of the software engineer to perform all the above activities (i.e. subjective processes). The proposed approach was still theoretical at the date of publication: no industrial case studies are reported to verify the applicability of the proposed approach and it is only limited to business software applications.

(Bellur et Vallieswaran, 2006) proposed a relational meta-model and an algorithm to check consistency on the proposed meta-model in Iterative Development Environments (IDE). The algorithm aim is to maintain the consistency between design representations and to maintain the traceability of design changes in code as well. The proposed relational meta-model checks for consistency by looking at the well-formedness of each design entity, and the relationships between the different design entities. Two problems were detected in using the UML notation for documenting system design, as follows:

- each diagram can be independently edited: this leads to inconsistency problems between different diagrams; and
- most of code generation tools in Interactive Development Environments use class diagrams only in the code generation process: this produces a system that reflects only one design perspective.

(Ouwerkerk et Abran, 2006) proposed an approach aimed to maintain inter-model consistency between scenario and class models by considering that these models are semi-formal, loosely coupled, and complementary: scenario model presents the external behavior

of a system and the class model presents internal functionality that is represented in a scenario model. A set of rule is proposed to develop consistent models by minimizing the overlap and cross-referencing information between the two models. On the other hand, the proposed approach is not tested, at the date of publication, with industrial requirements specifications.

(Salger, Engels et Hofmann, 2009) proposed a method called specification quality gate (QG-Spec) aimed to evaluate software requirement specifications: (Salger, Engels et Hofmann, 2009) suggested that inspections have to be balanced with techniques for constructive quality assurance in order to economically arrive at high quality SRS. However, in the discussion of the obtained results, (Salger, Engels et Hofmann, 2009) claimed that the method works fine in checking the completeness attribute but they also suggest the inclusion of customer to help fixing incompleteness and they present difficulties of this inclusion. Moreover, (Salger, Engels et Hofmann, 2009) mentioned that the application of the method relies on inspectors to make valid questionnaires to the project stakeholders which means that this method relies on the inspectors' knowledge and experience especially for the first two steps of the method. In other words, the method is subjective to the inspector experience.

(Knauss, Boustani et Flohr, 2009) proposed a quality model that aimed to objectively measure requirements quality: the hypothesis is that the quality of a software requirements specification strongly influences the probability of its project success. (Knauss, Boustani et Flohr, 2009) claimed that the quality of software requirements specifications can be measured based on an objective metrics: they suggested that if the quality of the requirements is below a certain quality threshold then the project is more likely to fail. In order to calibrate such threshold (Knauss, Boustani et Flohr, 2009) have analyzed forty (40) software projects based on the Goal-Question-Metric method and based on that analysis they have calibrated two thresholds as follows:

- lower threshold: projects have a SRS quality below this value are highly endangered; and
- higher threshold: projects have a SRS quality above this value are likely to succeed.

On the other hand, the proposed thresholds are not really representative because their analysis did not include industrial case studies where actual inconsistencies appear frequently but the analysis included forty (40) students' projects.

(Wang et al., 2009) proposed a use-case semi-automatic approach to the construction of feature models: this approach suggested to construct a set of feature models for individual applications (called application feature models – AFMs) in a software domain. Then, it is proposed to adjust and merge the set of application feature models to form a feature model for this domain (called domain feature model – DFM).

This approach provided a set of rules and algorithms to make the construction of AFMs and the construction of DFMs process automatically, as follows:

- construct a set of feature models for individual applications called application feature models from use cases of these applications with the support of a set of discovery rules;
- adjust the AFMs with the help of a set of adjusting rules and conflicts checking rules; and
- merge the set of adjusted AFMs to form a feature model for the domain called a domain feature model.

To derive features from use-cases: (Wang et al., 2009) proposed a structural description method for use-cases. On the other hand, the proposed approach of (Wang et al., 2009) has been tested using a simple case study that consists of only three (3) use-cases: this is currently a high-level of threat to validity on its applicability and usability. Furthermore, the meta-model of the use-cases does not consider the identification of the level of granularity of use-cases which are typically prepared the software engineers at early stage of the software development life cycle.

(Xu et al., 2011) proposed three-level use-case model to represent the functional requirements of software:

- function use-case level: consists of a set of function use-cases and a function use-case is defined as a method that is provided by the software system: it transforms the request of

the environment that interacts with the software system into the response of the software system;

- system use-case level: consists of a set of system use-cases and a system use-case is defined as a process of interaction between the software system and the environment or a finite sequence of function use-cases invoked by the environment; and
- business use-case level: consists of a set of business use-cases and a business use-case describes a set of system use-cases and their relationships with each other.

The authors in (Xu et al., 2011) classified the use-cases of a software system into three different classes but there is no identification of the level of granularity of each use-case in these classes.

(Ochodek et al., 2011) conducted a study aimed to evaluate the reliability of transactions identification in use-cases by four (4) transactions identification methods through conducting an experimental study with a group of one hundred and twenty (120) students who randomly formed four (4) sub-groups each of which applies one transaction identification method. The four (4) transactions identification methods are:

- method #1: counting stimuli-verbs in accordance to Robiolo and Orosco approach;
- method #2: identifying transactions based on Karner's UCP (Karner, 1993);
- method #3: identifying transactions based on the definition in UCP (Diev, 2006); and
- method #4: identifying transactions by using semantic transaction types.

The results of the experimental study showed that the choice of the method for transaction identification can have a visible impact on the values of the use-case-based functional size measurement and a set of counting rules should be developed to reduce the intra-method variability.

The definition of quality that was proposed by (Moody, 2005) is adopted by Marín et al. (2010): this definition uses a quality model to detect defects of conceptual models in Model-Driven Development (MDD) environments. The quality model suggested by (Marín et al.,

2010) is specified by means of the Extended Meta-Object Facility (EMOF) specification (Eclipse, 2010). In EMOF, a meta-model is represented by means of a class model where each class in the class model corresponds to a construct of the modeling language involved. As presented in (Marín et al., 2010) the quality model is comprised of the following:

- meta-model that contains a minimal set of conceptual constructs, their properties, and their relationships to allow complete specification of software products in the conceptual models of Model-Driven Development environments; and
- set of rules for the detection of defects in the conceptual model; which have been specified using OCL constraints (OMG, 2006).

Before elaborating in the two parts of the quality model, (Marín et al., 2010) proposed this definition: *"A meta-model is an artifact that is used to specify the abstract syntax of a modeling language; the structural definition of the involved conceptual constructs with their properties, the definition of relationships among different constructs, and the definition of a set of rules to control the interaction among the different constructs specified in such a meta-model"*.

The meta-model presented in (Marín et al., 2010) is designed to comprise a minimal set of conceptual constructs that should exist in order to provide a complete specification of a software product by means of a conceptual model. The specified conceptual model will be used to produce the software product using Model-Driven Development processes.

The conceptual constructs that allows for complete specification of the conceptual model are:

- the structural model;
- the behavioral model; and
- the interactional model.

The structural model is the meta-model construct that is designed to describe the static view of the software product to be produced. Normally, the structural model is represented by means of a class diagram which shows the common characteristics, semantics, and

constraints of a set of objects. In a class diagram, a class has a set of attributes: these attributes can be atomic or derived ones that obtain their values from other attributes. The behavior of a class is represented by mean of services: these services are categorized into events, transactions, and operations. Each service can be associated with a precondition that controls its execution.

Integrity constraints are typically applied in order to maintain all the objects of a class in a valid state. Classes can have relationships with other classes: relationships can be categorized into association, aggregation, composition, specialization, or agent relationship. More details about the specification of the behavior of classes are presented in the behavioral model.

The behavioral model is the meta-model construct designed to describe the dynamic view of the software product to be produced. The behavioral model specifies the behavior of each class specified in the structural model along with the interactions of the system objects.

The interactional model is the conceptual construct that is designed to describe the presentation and the dialogues of the software product to be developed: it is designed to present the static and dynamic aspects of a software product. These aspects are specified by means of “views” which corresponds to a set of interfaces and work as the communication point between agents and classes in the structural model (Marín et al., 2010).

2.4.2 Experimental studies on requirements quality

Other studies involved conducting experimental studies on requirements quality. For instance, (Anda, Sjøberg et Jørgensen, 2001) conducted an experimental study with three (3) groups of participants for a total of 139 undergraduate students: each group used one different guideline to construct a use-case model from an informal set of requirements specifications. Then, each participant in this study filled a questionnaire to provide feedback about the experiment. The results of the experimental study showed that using template-

based guidelines is easier, more useful and provides better quality models to the readers than guidelines that provide no details on how to document a use-case.

(Gomaa et Wijesekera, 2003) proposed an approach to identify and correct inconsistency and incompleteness across UML diagrams and more specifically in use-case diagrams, class diagrams, and sequence diagrams. The approach describes checking the consistency between multiple diagrams of software design using the Object Constraint Language constraints (OMG, 2006).

(Kuzniarz et Staron, 2003) presented inconsistencies found in student designs. These designs were produced in an introductory course to object oriented software development with UML. The student designs were checked for inconsistency using a set of rules of thumb which claimed to be used in future for more precise and formal definition of consistency in UML models.

(Lange et Chaudron, 2004) conducted interviews and surveys during a software project with practitioners in large-scale industrial organizations: the following problems are reported due to doubt about model completeness:

- uncertainty about accuracy and precision of estimates;
- miscommunication; and
- integration overhead.

The conducted interviews with industrial software engineers by (Lange et Chaudron, 2004) identified incompleteness of UML designs as a potential problem for subsequent stages of development in layers fashion in which software artifacts are presented in terms of their level of abstraction (decreases from top to bottom): the top layer represents the software requirements and these requirements are detailed by use-cases. The sequence charts are more detailed instantiations of use-cases. The relationship between classes and sequence charts means that an instantiation of the class occurs as object in a sequence chart. The internal

behavior of classes is described by state diagrams. Finally, the bottom layer is the implementation (i.e. the actual source code) (Lange et Chaudron, 2004).

(Lange et Chaudron, 2004) identified the concept of completeness from two perspectives:

- client perspective: starts with an idea that is transformed into user requirements: the client is interested in a product that exactly meets the user requirements; and
- software maker perspective: it is important for the software maker to develop the software in an economical way (i.e. low resource cost, short time-to-market) which enables the software maker to maximize profits.

(Lange et Chaudron, 2004) decomposed the concept of completeness to the sub-concepts of requirements completeness and UML modeling completeness. UML modeling completeness is decomposed into:

- well-formedness of each single diagram;
- consistency between diagrams; and
- completeness amongst diagrams.

(Lange et Chaudron, 2004) considered that a UML model is complete if for each element in the one diagram its expected counterpart in the other diagram is present. An inter-diagram incompleteness happens when there is an element in the overlapping part of two diagrams in the one diagram without matching counterpart in the other diagram.

(Leung et Bolloju, 2005) conducted a study aimed at understanding errors frequently committed by novice systems analysts in developing domain models using UML: the study reported results from analyzing class diagrams produced by (15) teams of novice systems analysts as part of e-business systems requirements specification using (Lindland, Sindre et Sølvberg, 1994) framework: (Leung et Bolloju, 2005) analyzed the quality of class diagrams in fifteen (15) project reports undergraduate students in an object-oriented analysis and design course: the authors identified a total of one hundred and three (103) in class diagrams.

The study of (Leung et Bolloju, 2005) included the analysis class diagrams of undergraduate students and no industrial case studies were included in the analysis. (Leung et Bolloju, 2005) considered that the involved project reports from the fifteen (15) teams of students are realistic representation of UML class diagrams produced by novice analysts.

(Lange et Chaudron, 2007) reported a multiple case study aimed to explore the level of defect occurrences in industrial UML modeling in a set of sixteen (16) industrial UML models: they explored the influence of factors such as model size, time pressure, developer's skill and application domain on model quality. (Lange et Chaudron, 2007) reported that the number of defects is alarmingly large in industrial UML models and prevention techniques such as modeling conventions, training, and tooling have to be adjusted to focus on common defect types. These authors found some indicators that affect defects occurrence such as time pressure, human factor, and quality assurance.

(Abu-Talib et al., 2008) presented the applicability of the COSMIC functional size measurement method in assessing hardware-software requirements allocation with an assumption that functional size measurement feedback will help software developers in their trade-off analysis when allocating functionality to software and hardware. On the other hand, the authors suggest a generic two-steps procedure to handle the ambiguity inherent in real-time software specifications but they do not identify such procedure or how it should be applied.

(Trudel et Abran, 2010) investigated the contribution of functional size measurers to finding defects in requirements: they conducted an experiment where the same requirements document was inspected by a number of inspectors and by a number of measurers. Most participants in the experiment had limited experience in both inspecting and measuring, the measurers have used COSMIC method to measure functional size and to find defects:

- results showed an increase in defect identification when both inspection and functional size measurement are used to find and report defects;

- results showed a value-added factor in terms of defects found when a measurer raised defects and issues while measuring functional size; and
- adding one measurer over an inspection team allowed the number of new critical and minor functional defects identified to rise between 17% and 58%, requiring 22% more effort than the individual checking effort.

An candidate explanation for the added-value results with measurers and the fact that their defects were 100% functional is that: the measurers were looking at specific pieces of information such as the conformity of the functional processes with the definition of a functional process and the clarity of the definition of the data movements and the data groups.

(España et al., 2010) aimed to assess the quality of functional requirements specifications using the Method Evaluation Model (MEM) as a theoretical framework: their focus is on the completeness and granularity of requirements models by defining metrics like degree of functional encapsulation, completeness with respect to a reference model, number of functional fragmentation errors, to measure the level of completeness. For comparison purposes, an experiment was carried out with seventeen (17) students to compare two requirements engineering methods, namely use-cases (Cockburn, 2000) and Communication Analysis (España, González et Pastor, 2009).

(España et al., 2010) suggested that an expert modeling committee analyze a given domain and agree on a model that strictly follows best practices in modeling and then measure the degree of functional encapsulations completeness and the degree of linked communications completeness with respect to a reference model (which is agreed by an expert modeling committee).

After that, (España et al., 2010) suggested assessing whether a functional requirements specifications document has an appropriate granularity level which allows determining the number of functional fragmentation errors and functional aggregation errors.

The experiment findings are:

- communication analysis allows obtaining requirements specifications with greater degree of functional encapsulation completeness and greater degree of linked communication completeness than use-cases;
- communication analysis allows obtaining requirements specifications with less functional aggregation errors than use-cases;
- communication analysis allows obtaining requirements specifications with less functional aggregation errors than use-cases;
- communication analysis was perceived as more useful than use-cases; and
- use-cases were perceived as easier to use than communication analysis.

(Bolloju et Sun, 2012) aimed to present the benefits of supplementing each use-case narrative with an activity diagram for analysts and clients during requirements gathering and analysis using one hundred twenty-seven (127) pairs of use case narratives and activity diagrams obtained from thirty-one (31) undergraduate student projects in a systems analysis course: these student projects represent systems from six (6) different industries, including banking, insurance, healthcare and trading. The results of the study showed that:

- process logic in activity diagrams is more complete and offers a greater degree of validity than that used in use-case narratives;
- quality of the process logic is not negatively affected by the length use-case narratives or complexity when they are used together to capture system requirements; and
- semantic quality of the process logic captured by the activity diagrams is superior to the quality of that captured by the use-case narratives.

On the other hand, (Bolloju et Sun, 2012) suggested to supplement use-case narratives with activity diagrams at early the phases of the software development life cycle when only high-level and incomplete requirements are available.

2.5 Chapter Summary

This chapter has presented an analysis of the literature related to quality of software requirements: research initiatives proposed quality models including meta-models and rules aimed to maintain high-level of software requirements consistency, correctness, completeness and un-ambiguity. Other research initiatives conducted experimental studies with groups of participants mainly in the academia to verify the usability of different requirements specification techniques. However, none of the proposed research initiatives took into consideration or even neglected the fact that software requirements specification are typically documented by software engineers at different levels of granularity except one research study by (Azzouz et Abran, 2004). The current practice in the software engineering industry is to accept incomplete and non-detailed software requirements specifications documents (i.e. to have portions of requirements documented at different levels of granularity) and without systematically handling this diversity by either the software engineers and to identify the level of granularity of these software requirements specifications subjectively using intuition, experience or opinion of the field experts. The analysis of the literature also showed that there exists a relationship between the quality of an SRS document and software project success and the complete identification and specification of software requirements is a valid strategy to reduce cost overruns of software projects.

CHAPTER 3

RESEARCH GOAL, OBJECTIVES AND METHODOLOGY

3.1 Introduction

The research project definition phase is one of the primary phases towards solving a research issue. It helps the principal researcher to ensure the validity of the research activities and the research results. (Ellis et Levy, 2008) adopt the 'divide and conquer' strategy in order to solve large and complex research issues: they consider that from a research standpoint, almost each research issue can be subdivided into smaller research issues and it is easier to address and resolve such subdivided research issues.

This chapter presents the research project definition phase including: the research motivation, the research approach, the research goal, the research objectives, the key inputs to this research project, the users of the research results and the research methodology.

3.2 Research Motivation

The motivation of this research project is to help software organizations and in particular projects managers and technical leaders to build more accurate effort estimation models by improving one of the inputs for the effort estimation process in order to improve the planning, the management, and the development of software at early phases of the software development life cycle.

3.3 Research Approach

The identification of the levels of granularity of functional requirements specifications is currently handled by the personnel involved in the approximation process of software functional size subjectively using intuition, experience or opinion of the field experts.

This challenge should be handled by the requirements engineers who are responsible for the elicitation of the functional requirements from the stakeholders of software development projects by following a systematic methodology to assign the functional requirements specifications scaling factors to rank their levels of granularity.

Next, the personnel involved in the approximation of software functional size use the functional requirements specifications – in which their levels of granularity is identified – to calculate an approximation of software functional size.

For further illustration of this research issue, an analogy from the architectural engineering discipline is presented next.

Architectural drawings (i.e. drawings of buildings and bridges…etc.) are drawn by architectural engineers and architectural professionals to construct a comprehensive architectural proposal of a building project. These drawings are drawn in accordance to predefined standards and customs (e.g. sheet sizes, units of measurement and scales, annotation and cross referencing) where the level of detail of each component in these drawings is often expressed by using internationally standardized scale units.

On the other hand, in the software engineering discipline, there is no standardized identification or application of such scale factors in the software requirements specifications documents. Therefore, this leads to the calculation of less accurate approximation of the functional size of software.

3.4 Research Goal

Obtaining more accurate effort estimation models for software development projects helps the software projects managers in preparing more realistic projects charters. The goal of this research project is to improve one of the inputs of the *a priori* effort estimation process and in particular the functional size approximation of software development projects. The main

objective of this research project is to design a framework that assign scaling factors for identifying the levels of granularity of functional requirements specifications of software which are typically documented at different levels of granularity at the early stages of the software development life cycle.

3.5 Research Objectives

To achieve the main goal of this research project, the following specific research objectives have been selected for this thesis and must be achieved:

1. an investigation of the impact (and limitations) of an existing functional size approximation method from the literature (i.e. the *E&Q* COSMIC technique) on the accuracy of the results of the functional size approximation.
2. a proposed new framework to assign scaling factors to early functional requirements specifications to identify and rank their levels of granularity, including:
   (a) a set relevant concepts and their relationships that need to be collected by the requirements engineers to allow full functional specification of software functional requirements;
   (b) a set of scaling factors to rank the levels of granularity of functional requirements specifications;
   (c) a meta-model that captures the relevant concepts with their relationships and the defined scaling factors; and
   (d) a set of criteria to identify the levels of granularity of functional requirements specifications.
3. the proposed scaling factors framework verified for usability and applicability.
4. an investigation of the impact of using the framework on the accuracy when the functional size is approximated using the statistical table of the *E&Q* COSMIC technique.

3.6 Research Inputs

The main inputs for this research project are:

- (ISO19761, 2011): an international standard for software functional size measurement.

- (COSMIC, 2011): Guideline for assuring the accuracy of measurements - COSMIC v. 3.0.1, Common Software Measurement International Consortium.
- (Desharnais, Kocaturk et Abran, 2011): Using the COSMIC Method to Evaluate the Quality of the Documentation of Agile User Stories.
- (Vogelezang et Prins, 2007): Approximate size measurement with the COSMIC method: factors of influence.
- (COSMIC, 2007): Advanced & Related Topics COSMIC v 3.0, Common Software Measurement International Consortium.
- (Azzouz et Abran, 2004): A proposed measurement role in the Rational Unified Process and its implementation with ISO19761 – COSMIC.
- (IEEE, 1998): IEEE-830 Recommended practice for software requirements specifications.
- research studies (2000 - 2012) on software functional size measurement and quality of software functional requirements.

It is worth mentioning that the main reasons for choosing the COSMIC method as one of the primary inputs in this research project are:

- the COSMIC measurement method is standardized by the International Organisation for Standardization (ISO19761, 2011);
- it represents the 2nd generation of software functional size measurement methods; it is getting attention from the software measurement community at the industrial and academic levels over the last decade; and
- it measures the software functionality from the software user's perspective at a relatively early stage of the software development life cycle.

3.7 Overview of Research Methodology

To achieve the objectives of this research project, the literature related to approximation of software functional size and related to quality of software functional requirements have been identified and analyzed in order to design the different phases of the research methodology.

This section presents an overview of the research methodology designed by the principal researcher to achieve the research objectives in eight phases – figure 3.1.

Phase 1: Literature review

Phase 1 of the research methodology consists of identifying the research issues of the software functional size to approximation, analyzing the literature related to early approximation of software functional size, and the literature related to the quality of software functional requirements and the presence of various levels of granularity within software requirements specification documents.

Phase 2: Experimentation of an existing COSMIC approximation method

Phase 2 of the research methodology will include an experiment evaluate the reproducibility and accuracy of the approximation results using only the concepts and the rules of a functional size approximation technique proposed in the literature, namely the *E&Q* COSMIC technique with one group of industry practitioners.

Phase 3: Design of the scaling factors framework

Phase 3 of the research methodology will consist of identifying the basic building blocks for a scaling factors framework; this includes identifying the relevant concepts needed for the specification of the functional requirements of software and defining the format and the type of the scaling categories. This phase also consists in designing a meta-model to capture the identified relevant concepts with the defined scaling categories. Further, this phase consists in designing criteria to identify the level of granularity of the functional requirements of software development projects.

Phase 4: Certification from the Committee for Ethics in Research

Phase 4 of the research methodology will consist of applying for certification from the Committee for Ethics in Research (CÉR) at École de technologie supérieure. The CÉR examines and approves the ethical aspects in the research projects that involve human participants in the experimental studies. The ethical aspects including the mode of

recruitment of participants, respect for freedom of participation, the risks and benefits of this research project, and mechanisms to ensure the confidentiality of the collected data (See Appendix I on the CD attached to this thesis for the certificate of CÉR).

Phase 5: Verification of the usability of the scaling factors framework

Phase 5 of the research methodology will consist of the verification of the usability of the scaling factors framework with a variety of groups of practitioners in the software engineering industry. It will involve conducting an experimental study with three (3) groups of industry practitioners to apply the scaling factors framework to the same case study.

Phase 6: Definition of scaling factors

Phase 6 of the research methodology will define a set of scaling factors to be assigned to the software requirements specifications – in which their levels of granularity is identified using the set of criteria – to rank their levels of granularity using the measurement unit of the COSMIC measurement method (i.e. COSMIC Function Point – CFP).

Phase 7: Verification of the applicability of the scaling factors framework

Phase 7 of the research methodology will consist of the verification of the applicability of the scaling factors framework to a variety of case studies. It involves conducting an experimental study with case studies that represent software applications that are different in software type, business objective, context, and software functional size.

Phase 8: *E&Q* approximation with partial support of the scaling factors framework

This phase of the research methodology will include the approximation of the functional size of four (4) case studies using *E&Q* COSMIC technique after identifying the levels of granularity of the functional requirements specifications of the case studies using the scaling factors framework. It will also include a comparison of the functional size approximation calculated using *E&Q* COSMIC technique and the functional size approximation calculated using the interval scaling factors.

3.8 Detailed Research Methodology

Phase 1: Literature review

The objective of this phase was to develop an in-depth understanding of the scaling factors issue. This step is focused on analyzing the literature related to early approximation of software functional size and the literature related to the quality of software functional requirements. The aim of this step is to build an in-depth understanding of the literature contributions, the challenges encountered, and their shortcomings.

Phase 2: Experimentation of an existing COSMIC approximation method

This step will consist of an experiment to aimed to evaluate the reproducibility and accuracy of the approximation results using only the concepts and the rules of the *E&Q* COSMIC technique: this will be achieved by asking a group of practitioners with significant experience in the software engineering industry to apply the *E&Q* COSMIC technique to a set of early functional requirements specifications.

Step 2.1 Objective and purpose experimentation

The objective of this experiment is to evaluate the reproducibility and accuracy of the approximation results of one of the *E&Q* techniques using the information actually available to the practitioners that is without the preliminary steps not available to the practitioners and to the industry in general: the participants in this experiment will be asked to classify a set of software requirements specifications in accordance to their level of granularity into the different *E&Q* COSMIC functional components.

Step 2.2 Context and scope

This step will focus on the identification of the case study that is suitable for this experimental study: the case study that has been selected presents real-time software that is used for usability testing.

Step 2.3 Experiment preparation

This step will focus on preparing the experimental settings for experimentation including the experiment material such as the case study, and the participation consent form. The experimental settings will also include the call-for-participation, pre-experiment training session and conducting a pilot test of the experiment with the help of an independent expert in order to identify and eliminate any challenges prior to the experimental session.

Step 2.4: Experimentation and analysis

This step will focus on conducting the experiment with the participants who will volunteer to participate in the experimental study. The objective of this step is to observe the impact of applying an early functional size approximation technique without using the scaling factors framework or any other guideline to identify the correct level of granularity of the requirements specifications presented in the case study.

Phase 3: Design of the scaling factors framework

The objective of this phase is to design the scaling factors framework to identify the levels of granularity of functional requirements specifications of software development projects. This phase of the research methodology will consist of the following steps:

Step 3.1: Identification of the relevant concepts

This step will focus on the identification and the analysis of the relevant concepts that the requirements engineers need to consider to allow for a full-functional specification of the functional requirements at the early stages of software development projects.

Step 3.2: Definition of the scaling factors

This step will focus on the definition of the type and the format of the scaling factors that will be assigned by the scaling factors framework to the functional requirements of software development projects.

Step 3.3: Design of the meta-model

This step will focus on the design of a meta-model to capture the identified concepts and the defined scaling factors: this step will include the selection an appropriate modeling language for the design of the meta-model and building the meta-model hierarchy.

Step 3.4: Design of criteria

This step will focus on the design of criteria to be applied by the requirements engineers to identify the level of granularity of early versions of the functional requirements specifications on the early stages of the software development life cycle.

Phase 4: Certification from the committee for ethics in research

The objective of this phase is to obtain a certification from the committee for ethics in research at École de technologie supérieure – Université du Québec in order to acknowledge and approve the ethical aspects in this research project. The committee for ethics in research must ensure that the participation of the participants in the experimental studies is fully ethical including mode of recruitment, freedom of participation, the non-existence of risk due participation, the confidentiality of participation and the collected data in the experimental studies. This phase of the research methodology consists of the following steps:

Step 4.1: Application for certification

This step will focus on the presentation of all the necessary documentation for the committee for ethics in research in order to obtain their approval to conduct the experimental studies: this includes filling the application form, copy of the experiment material such as the participation consent form, the number of participants in each experimental study and any existence of conflicts of interests and how these conflicts are mitigated or avoided and the place of preserving the collected data.

Step 4.2: Reporting to the committee for ethics in research

This step will focus on providing a full report to the committee for ethics in research after conducting the experimental studies, and completing the analysis of the data collected during the experimental studies: it includes reporting of any change on the mode of experimentation

or the research methodology, and the actual number of participants who actually participated in the experimental studies.

Phase 5: Verification of the usability of the scaling factors framework

The objective of this phase is to verify that the scaling factors framework is usable by a representative variety of industry practitioners with different levels of experience in the software engineering field: they should have been involved in different stages of the development life cycle of software projects. This phase will include conducting three (3) experimental sessions to apply the scaling factors framework with the same case study by three (3) different groups of industry practitioners.

Step 5.1: Objective and purpose of the experimentation

This experimental step is to verify the usability of the designed scaling factors framework: the participants in this experimental study will be asked to assign scaling factors to a set of non-complete, non-detailed software requirements specifications at the early stages of the software development life cycle by using the scaling factors framework in order to identify the levels of granularity of these requirements specifications.

Step 5.2: Context and scope

This step will focus on the identification and selection of the case study that is suitable for this experimental study: the case study that has been selected presents a management information system. The case study that will be selected has to be easy to understand in order minimize the learning curve of the participants in understanding the case study and keep their concentration on understating the concepts that the scaling factors framework has to offer: this is due to the limited time the participants will have to apply the concepts the scaling factors framework on the requirements specification presented in the case study.

Step 5.3: Experiment preparation

This step will focus on preparing the experimental settings for experimentation including the experiment material such as the scaling factors framework, the case study, and the

participation consent form. The experimental settings will also include the call-for-participation, pre-experiment training session, and conducting a pilot test of the experiment with the help of an independent expert in order to identify and eliminate any challenge prior to the experimental session.

Step 5.4: Experimentation and analysis

This step will focus on conducting the experimental study by the three (3) groups of participants who will volunteer to participate in this experimental study.

Phase 6: Definition of interval scaling factors

The objective of this phase is to define a set of interval scaling factors to rank the levels of granularity of the software requirements specifications using the measurement unit of the COSMIC measurement method (i.e. COSMIC Function Point – CFP). The levels of granularity of software functional requirements specifications – at this stage – is already identified using the set of criteria designed in Step 3.4.

Step 6.1: Analysis of the proposed interval scaling factors in the literature

This step will focus on analyzing the interval scaling factors proposed in the literature in order to identify the appropriate strategy to follow in the definition of the interval scaling factors of the scaling factors framework.

Step 6.2: Define the interval scaling factors

This step will focus on the definition of the interval (numerical) scaling factors that rank the levels of granularity of the functional requirements specifications using the measurement unit of the COSMIC measurement method. This step is to transform the set of ordinal scaling factors into a set of interval scaling factors.

Phase 7: Verification of the applicability of the scaling factors framework

The objective of this phase is to verify that the scaling factors framework is applicable to a variety of sets of software requirements specifications which represent software applications

that are different in software type, business objective, context, and software functional size: the principal researcher will select four (4) case studies that achieve these criteria. Another objective is to observe the impact of using the scaling factors framework prior to the functional size approximation process.

Step 7.1: Objective and purpose experimentation

The objective is to verify the applicability of the scaling factors framework with variety sets of case studies that represents software requirements specifications that are documented as observed in the early phases (typically non-complete and non-detailed requirements specifications) of the software development life cycle. Another objective includes observing the impact of identifying the correct levels of granularity of the requirements specifications on the approximation of functional size.

Step 7.2: Context and scope

This step will focus on the identification and selection of the case studies that are suitable for this experimental study: the case studies must present software applications from both business application domain and real-time domain and differ in their business objective, context.

Step 7.3: Experiment preparation

This phase will focus on preparing the case studies selected for experimentation by the principal researcher. This includes measuring their functional size using the international standard for functional size measurement: ISO19761 – COSMIC and modifying the functional specifications of the case studies in order to adapt them to the objective of this experimental study.

Step 7.4: Experimentation and analysis

This step will focus on conducting the experimental study by the principal researcher in order to observe the impact of applying the scaling factors framework to identify the correct levels of granularity of the requirements specifications presented in the case studies.

Phase 8: *E&Q* approximation with the partial support of the scaling factors framework

The objective of this phase is to illustrate the value added of using scaling factors framework over using the *E&Q* COSMIC technique in approximate sizing.

Step 8.1: Approximation of the functional size using the *E&Q* COSMIC technique

This step will focus on calculating an approximation of the functional size of four (4) case studies by applying the rules and concepts of the *E&Q* COSMIC technique from the literature. The levels of granularity of the functional specifications of the four (4) case studies are already identified in the experimentation Phase 7.

Step 8.2: Comparison and analysis

This step will verify whether or not there is value added in using the interval scaling factors of the framework on the accuracy of the functional size approximation over using statistical table of the *E&Q* COSMIC technique.

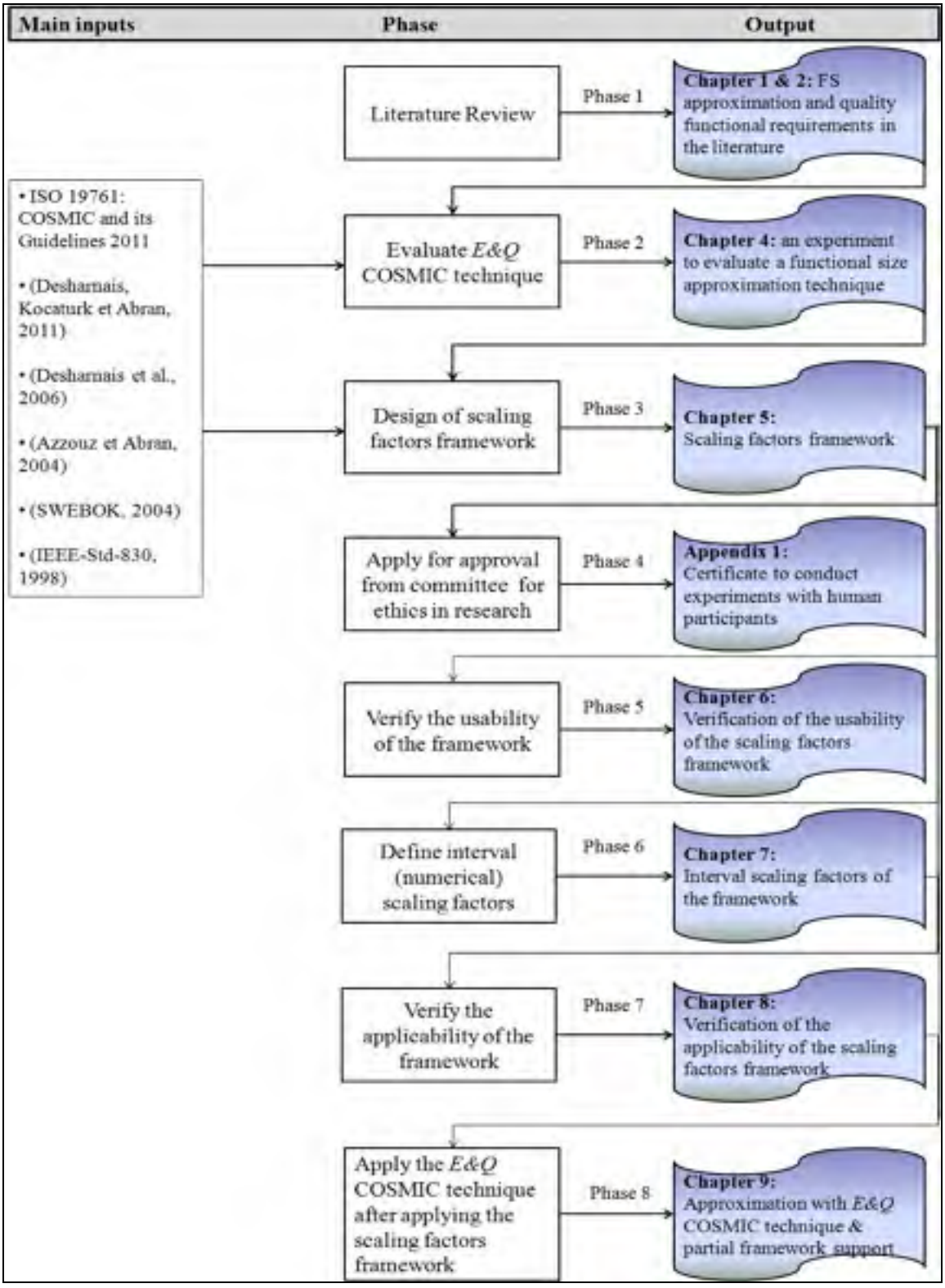

Figure 3.1 Overview of the research methodology

CHAPTER 4

AN EXPERIMENT TO EVALUATE A FUNCTIONAL SIZE APPROXIMATION TECHNIQUE

4.1 Introduction

This chapter presents an experiment with one group of industry practitioners aimed to evaluate a functional size approximation technique proposed in the literature, namely the *E&Q* COSMIC technique: this will be achieved by conducting an experiment aimed to evaluate the reproducibility and accuracy of the approximation results using only the concepts and the rules of the *Early & Quick* COSMIC technique by one (1) group of practitioners in the industry.

In the ISO International vocabulary of basic and general terms in metrology (ISO, 1993), reproducibility and accuracy are defined as follows:

- reproducibility, as a condition of measurement: *"condition of measurement, out of a set of conditions that includes different locations, operators, measuring systems, and replicate measurements on the same or similar objects"*; and
- accuracy, as applied to measurement: *"closeness of agreement between a measured quantity value and a true quantity value of a measurand"*.

The *Early & Quick* sizing techniques have been proposed to derive an early approximation of software functional size when only high-level, incomplete requirements specifications are available. In the literature, there is still a lack of research to evaluate the performance of approximation sizing methods built based on ISO standards, such as the *Early & Quick* sizing techniques (Meli, 1997), (Conte, Iorio et Santillo, 2004).

It is worth mentioning that two (2) preliminary steps are recommended in (COSMIC, 2007) for the usage of the *E&Q* COSMIC technique (Conte, Iorio et Santillo, 2004). However, no

details are provided in (COSMIC, 2007) to the researchers and practitioners on how to apply these steps in practice:

- identification of the levels' of granularity of requirements specifications; and
- identification and usage of size scaling factors.

The two (2) preliminary steps recommended in (COSMIC, 2007) were not taken into account in the experiment design due to the unavailability of related guidelines from the literature after fifteen years of the initial publication of the *Early & Quick* technique (Meli, 1997) and eight years after the publication of the COSMIC variant (Conte, Iorio et Santillo, 2004).

This chapter is organized as follows: Section 4.2 presents the *Early & Quick* techniques; Section 4.3 presents the context of the experiment designed to evaluate the reproducibility of the *Early & Quick* COSMIC technique. Section 4.4 presents the experimental results. Section 4.5 presents validity threats. Finally, the chapter summary and discussion are presented in Section 4.6.

4.2 The Early & Quick Techniques

The *Early & Quick* technique was initially published in 1997 for the original Function Points Analysis sizing method (DPO, 2007). As the COSMIC measurement method (ISO19761, 2011) got adopted in 2003 as an international standard for measuring the functional size of software, it became a necessity to generalize, and extend the initial design of the *Early & Quick* approximation technique to the COSMIC measurement method. The initial design of the *Early & Quick* COSMIC technique was proposed in (Meli, 2000), and after that, the *Early & Quick* COSMIC was generalized by (Conte, Iorio et Santillo, 2004).

In this context, the term *"Early"* refers to the need to obtain functional size approximation before a significant portion of the software requirements is detailed enough for precise measurement, and the term *"Quick"* means that typically such size approximation must be

obtained quickly, since they must be provided to management within a short time, in spite of the obvious constraints.

The *Early & Quick* techniques (Meli, 2000), (Conte, Iorio et Santillo, 2004) define a set of concepts and procedures which combine various functional size approximation approaches to derive an approximation for a software system's functional size. It classifies functions (functional processes & data groups) in an analogical and an analytical fashion.

The *E&Q* techniques provide the opportunity to utilize different levels of details of the software during the functional size approximation process. Therefore, the total amount of functional size uncertainty – within a range of values (minimum, most likely, and maximum) – will be the weighted sum of the uncertainty values of individual components. This chapter evaluates the reproducibility of one specific version of the *Early & Quick* techniques that is the *Early & Quick* COSMIC (*E&Q* COSMIC) (Conte, Iorio et Santillo, 2004).

Table 4.1 lists the various steps of these functional size approximation techniques that use analytically and analogically derived tables.

Table 4.1 Steps of the E&Q techniques
source: (Meli, 2000), (Conte, Iorio et Santillo, 2004)

Technique	**Explanation**
Classification by analogy	Similarity by size and/or overall functionality between new and known software objects.
Structured aggregation	Grouping of a number of lower level software objects in one higher level software object.
No given function/data correlation	Data and transactional components assessed autonomously.
Multilevel approach	No discarding of details, if available – no need for details, if unavailable.
Use of derivation tables	Each software requirement at every detail level is assigned a size value, based on analytically/statistically derived tables.

An *E&Q* functional size approximation starts with the breakdown of the structure of the software system under study, an example of which is shown in figure 4.1. The figure depicts the elementary functional processes, logical data groups and their aggregations representing different levels of detail. These heterogeneous levels of knowledge make it possible to take advantage of all the available information. In other words, the *E&Q* techniques enable the use of all the available non detailed information in the functional size approximation process. The elementary functional processes can be grouped into 'small', 'medium', or 'large' typical and general processes. General processes can be grouped into 'small', 'medium', or 'large' macro processes. On the other hand, the elementary logical data groups can be grouped into multiple data groups.

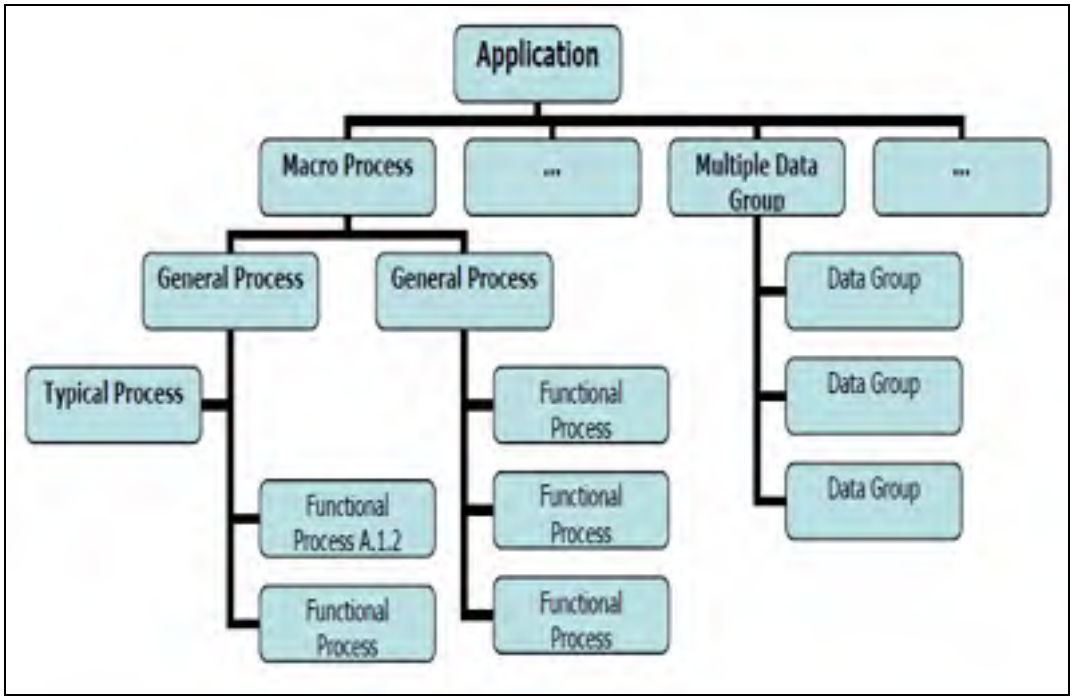

Figure 4.1 Functional hierarchy in the *Early & Quick* techniques – an example
source: (Conte, Iorio et Santillo, 2004)

It is worth mentioning that the functional processes in the *E&Q* IFPUG technique corresponds to the elementary processes of the standard IFPUG method: External Input (EI), External Output (EO), and External Query (EQ). On the other hand, the functional processes

in the *E&Q* COSMIC technique correspond exactly to the functional process of the standard COSMIC method and without distinction in their type.

Typically, the root is the highest level in the hierarchy (i.e. the application level), and lower levels stem from that root, based on the number of the software artifacts in the system under study. The method is applied down through the levels until the approximator decides that it is not useful to proceed with further decomposition (i.e. at the functional process level). Table 4.2 provides the descriptions and acronyms of the functional levels in the *E&Q* techniques:

- a functional process (FP) represents the smallest software process with autonomy and significance and corresponds to the functional process in the standard COSMIC method;
- a general process (GP) consists of a set of two or more average functional processes;
- a typical process is a particular case of a general process and typically consists of a set of the most frequently occurring operational transactions;
- a macro process (MP) con set of two or more average general processes;
- a logical data group (LDG) represents a group of logical data attributes; and
- a multiple data group (MDG) consists of a set of two or more logical data groups.

Table 4.2 The *Early & Quick* functional levels
scource: (Meli, 2000), (Conte, Iorio et Santillo, 2004)

Functional Level	**Brief Description**
Macro Process (MP)	The MP can be likened to a relevant subsystem, or even a bounded application, of an overall Information System.
General Process (GP)	It can be likened to an operational subsystem, which provides an organized, comprehensive response to a specific application goal.
Typical Process (TP)	It can be found in two "flavors": CRUD (Create, Retrieve, Update, and Delete), or (CRUD plus List, and Report).
Functional Process (FP)	It FP allows the user to achieve a unitary business objective at the operational level.
Multiple Data Group (MDG)	Its size is evaluated based on the approximated quantity of included LDGs.
Logical Data Group (LDG)	represents a conceptual entity that is functionally significant as a whole for the user.

It is worth mentioning that all the functions provided by the application must be at the leaf level, since there is no explicit functionality at higher levels of the hierarchy. Therefore, a functional approximation of all the leaves provides a bottom-up approximation of the whole tree (i.e. the software application).

The *E&Q* techniques assign a set of size values (minimum, most likely, and maximum) to each leaf in the hierarchy, based on the analytical and the analogical tables mentioned earlier. Then, these size values are summed to provide the overall approximation result (minimum, most likely, and maximum).

Table 4.3 *Early & Quick* COSMIC components' ranges and numerical assignment ranges source: (Conte, Iorio et Santillo, 2004)

Type	**Ordering**	**Ranges/COSMIC Equivalent**	**Minimum CFP**	**Most Likely CFP**	**Maximum CFP**
Macro Process	Small	2-4 General Processes	120	285	520
	Medium	4-6 General Processes	240	475	780
	Large	6-10 General Processes	360	760	1300
General Process	Small	6-10 Functional Processes	20	60	110
	Medium	10-15 Functional Processes	40	95	160
	Large	15-20 Functional Processes	60	130	220
Typical Process	Small	CRUD; and CRUD + List	15.6	20.4	27.6
	Medium	CRUD; CRUD + List; CRUD + List + Report	27.6	32.3	42
	Large	CRUD; CRUD + List; CRUD + List + Report	42	48.5	63
Functional Process	Small	1-5 Data movements	2	3.9	5
	Medium	5-8 Data movements	5	6.9	8
	Large	8-14 Data movements	8	10.5	14
	Very Large	14+ Data movements	14	23.7	30

It is worth mentioning that the IFPUG *E&Q* technique assigns numerical size values to logical data groups and multiple data groups, whereas, the *E&Q* COSMIC technique identify 'Objects of interest', but does not assign them any numerical size value. Table 4.3 reports

both the component ranges and the numerical assignments for the *Early & Quick* COSMIC approximation technique.

A reference manual for approximating function points at early phases of the software development life cycle using of the *E&Q* FPA technique is documented in (DPO, 2007). This manual describes the *E&Q* FPA technique without mentioning the need for any other guidelines for its application. More specifically, the goals of this reference manual (DPO, 2007) are:

- To provide an exhaustive and clear description of the FPA variant of the *E&Q* technique (i.e. *E&Q* FPA); and
- To promote comprehensive and homogenous application of *E&Q* FPA technique by providing a guideline to approximate function points at early phases of the software development life cycle.

The authors of this reference manual (DPO, 2007) mention that it was designed to be applied by practitioners with 'average' to 'good' knowledge of function points standard counting (i.e. the standard IFPUG method) (IFPUG20926, 2009). However, a detailed knowledge of the function points standard counting is not needed since the practitioners the *E&Q* IFPUG technique do not include any of the standard IFPUG rules and practices.

4.3 The Experiment Design

This section presents the eight (8) steps performed by the principal researcher to design experiment - see figure 4.2. These steps are explained in the next sections.

4.3.1 Experiment purpose & objective

The experiment purpose is to evaluate one of the *E&Q* functional size approximation techniques (i.e. *Early & Quick* COSMIC technique). The experiment specific objective is to evaluate the reproducibility and accuracy of the approximation results using only the

concepts and the rules of the *Early & Quick* COSMIC technique by a group of (12) practitioners in the industry. In this experiment, *'reproducibility'* refers to the degree of closeness between the results of functional size approximation calculated by different approximators on the same case study and *'accuracy'* refers to the level of closeness between the results of functional size approximation calculated by different approximators of the case study calculated as described below.

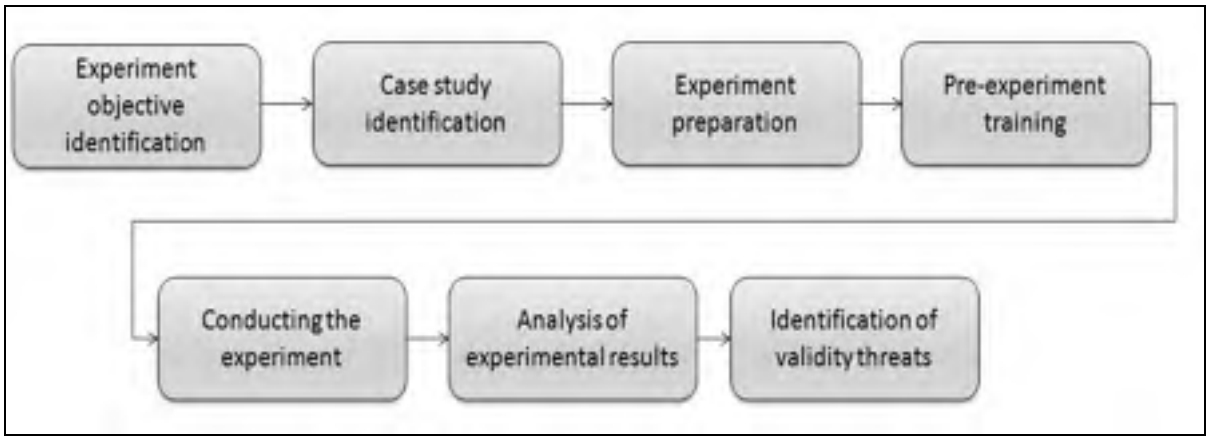

Figure 4.2 The experiment design steps

The experiment is conducted without taking into consideration the two (2) preliminary steps that are recommended in (COSMIC, 2007) for the usage of the *E&Q* COSMIC technique (Conte, Iorio et Santillo, 2004). This is mainly due to the unavailability of related guidelines from the literature after fifteen years of the initial publication of the *Early & Quick* technique (Meli, 1997).

4.3.2 Identification of the case study

The case study document from (Trudel et Lavoie, 2008) selected for the experiment is written in accordance to the UML 2.0 specifications (Arlow et Neustadt, 2005) and IEEE-Std-830 (IEEE, 1998) in terms of content and structure. The original case study document from (Trudel et Lavoie, 2008) consists of 18 pages of textual specifications, divided into three main sections:

- section 1 provides introductory information, including background information, software purpose and scope, software objectives, and references;
- section 2 provides a high-level description of the software to be developed, a list of the software functionality and features, the characteristics of the users, constraints, and assumptions; and
- section 3 provides the software functional and non-functional requirements, along with the user interfaces, the hardware interfaces, and the software prototype.

The functional size of the original document of this case study had been previously measured by a team of measurement experts using the international standard for software functional measurement: COSMIC (ISO19761, 2011). The measurement experts had an average of fifteen (15) years of industrial experience, were all COSMIC Certified Entry Level practitioners (Trudel, 2012), were experienced in functional size measurement, and were active members of the COSMIC Measurement Practice Committee.

Table 4.4 presents the functional size calculated by this team of experts and the average functional of 79.3 CFP. The differences in the functional size measurement results of each individual measurer are due to measurement assumptions made by the four (4) experts: the existence of the differences in the results does not mean that there were measurement errors, but it presents the different flavors of functional behavior – due to the assumptions – which could have been made by distinct development teams at the development phase of the software (Trudel, 2012).

Table 4.4 Functional size of the original case study document measured by the team of experts - source: (Trudel, 2012)

Measurer code	**Measured functional size**	**Average functional size**
Expert #B1	81	79.3 CFP
Expert #B2	71	
Expert #B3	68	
Expert #B4	97	

The original case study document from (Trudel et Lavoie, 2008) describes the functionality of the software system based on fifteen (15) use-cases that specify software system functionality in textual form. Table 4.5 presents the reference classification of the functional components of the software system by applying the *E&Q* COSMIC technique prepared by the principal researcher:

- eight (8) use-cases are classified as 'small' functional processes;
- five (5) use-cases are classified as 'medium' functional processes; and
- two (2) use-cases are classified as 'large' functional processes.

Table 4.5 The reference classification of the functional components of the case study selected for the experiment

Macro Process			**General Process**			**Typical Process**			**Functional Process**				**Approximation of software functional size (min, most-likely, max)**
S	*M*	*L*	*S*	*M*	*L*	*S*	*M*	*L*	*S*	*M*	*L*	*V. L.*	
-	-	-	-	-	-	-	-	-	8	5	2	-	(57 CFP, 87 CFP, 108 CFP)

On the right-hand side of table 4.5 is the approximation of the software functional size in CFP by applying the *E&Q* analytical/statistical table (i.e. table 4.3): the functional size approximation ranges (Min: 57 CFP, Most likely: 87 CFP, Max: 108 CFP).

4.3.3 Experiment preparation

This activity consisted of three sub activities: materials preparation, pilot testing, and call for participation, as follows:

Materials preparation:

Prior to the experiment session, the *E&Q* COSMIC technique was reviewed by the principal researcher in order to provide a description of the concepts and rules, and a procedure for applying the technique. The experiment materials (See Appendix II on the CD attached to this thesis) included:

- a description of the *E&Q* COSMIC technique;

- the case study document;
- a defined set of rules; and
- a defined set of participant roles.

The original case study document (Trudel et Lavoie, 2008) was considered to be detailed and complete, and it specifies in detail the functionality that has to be delivered by the software system. Therefore, it allows a standardized functional size measurement method to be used to obtain an accurate measurement of software system functional size. For the purpose of the experiment, the case study document was modified as follows:

- six (6) use-cases were kept 'as is' (i.e. without any modification of their specifications);
- four (4) use-cases were partially modified, by removing portions of specifications; and
- five (5) use-cases were completely modified by removing use-cases specifications entirely.

Pilot Testing:

A preliminary run of the experiment by the principal researcher and the independent expert (i.e. Expert #B5) - who verified the reference classification - to identify early on the potential challenges in the experiment procedures, including:

- the applicability of the SRS to the experiment, in terms of scope and objective;
- an estimate of the time required to conduct the experiment;
- usability of the data collection forms to be used by the participants in the experiment; and
- Verify the correctness of the reference classification of the functional components of the software system (see table 4.5) by asking the independent expert (i.e. Expert #B5) to conduct the experiment activities.

Call for Participation:

This experiment was stage as part of the 2nd International Symposium in Software Engineering Management (ISSEM 2011). Twelve participants volunteered to conduct the experiment. The industrial experience profile of the twelve (12) participants involved in the experiment is presented in figure 4.3 in accordance to their industrial experience in software

engineering topics: the participants had an average of nine (9) years of experience in software requirements analysis & modeling, software development, software documentation, software quality assurance, and software project management. It can be also observed from figure 4.3 that participants (P1) to (P8) have an average of 12 years of industry experience, while participants (P9) to (P12) have very limited industry experience. In summary, two thirds of the participants had significant industry experience.

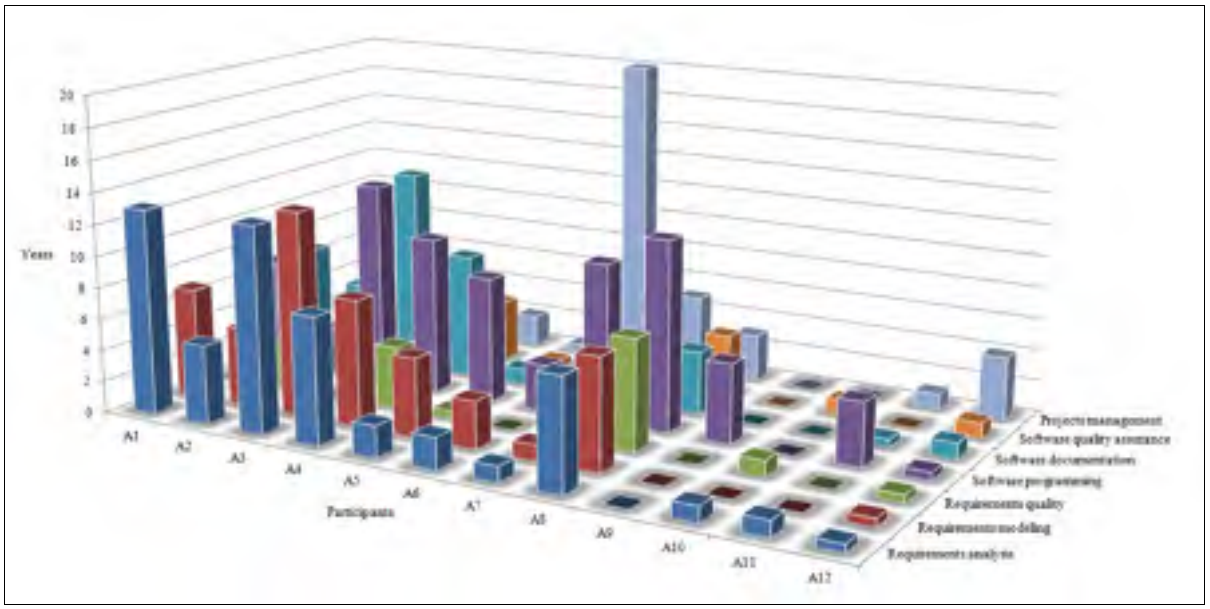

Figure 4.3 The industrial experience (in years) of the participants in the experiment

4.3.4 Pre-experiment training

The participants in the experiment were given a one-hour training session, to familiarize them with the *E&Q* COSMIC technique, the rules to follow, and the roles that would govern the participants' behavior during the experiment. The participants were then given thirty minutes to read the case study document.

4.3.5 Conducting the experiment

The participants were given one (1) hour to:

A. Classify the set of software requirements specifications as *E&Q* COSMIC functional components in accordance to their level of granularity; and then
B. use the statistical table of the *Early & Quick* COSMIC technique (i.e. Table 2) to calculate an approximate functional size of the software system presented in the modified SRS document (see figure 4.4).

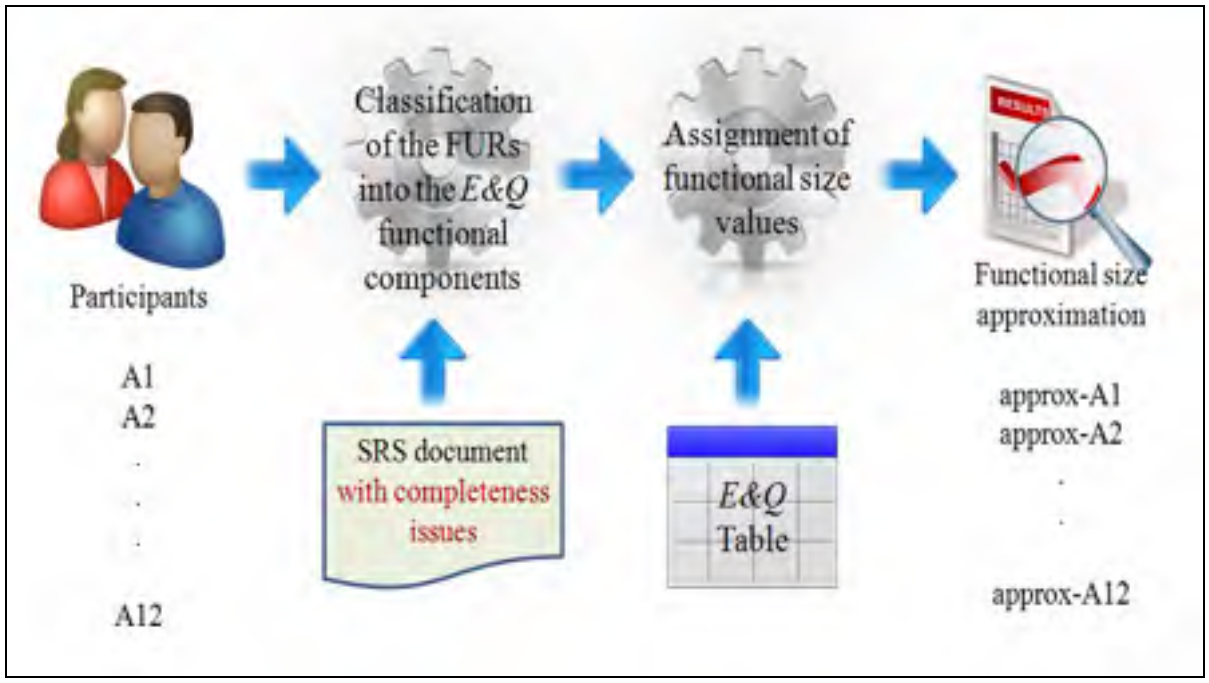

Figure 4.4 An overview of the activities of the participants in the experiment

The following experimental data were to be captured on forms (See Appendix II on the CD attached to this thesis) designed for this purpose:

- software process types: Functional, General, Typical, or Macro;
- total number of software processes for each process type;
- total functional size for each process and total functional size of the software system; and
- total effort required to approximate the functional size.

4.4 The Experimental Results

This section presents the evaluation of the reproducibility of the functional size approximations calculated by the participants in the experiment, as well as, the evaluation of the accuracy of their functional size approximations with relative to the reference functional size of the case study (see table 4.4).

4.4.1 Descriptive data from the experiment

To explore whether or not the experience of the participants had an impact on the results of the experiment, the results are presented in two groups:

- results of the 8 participants in table 4.6 with an average of 12 years of industry experience;
- results of the 4 participants in table 4.7 with an average of 1 year of industry experience.

Table 4.6 Classifications of the participants with an average (12) years of experience

Participant Code	**Macro Process**			**General Process**			**Typical Process**			**Functional Process**				**Functional size (min, most-likely, max)**	**Effort (minutes)**
	S	*M*	*L*	*S*	*M*	*L*	*S*	*M*	*L*	*S*	*M*	*L*	*V. L.*		
A1	-	-	-	8	-	-	-	-	-	12	17	-	18	(521, 1071, 1616)	55
A2	-	-	-	7	3	-	-	-	-	-	-	-	-	(250, 705, 1250)	50
A3	-	-	-	4	2	-	4	-	-	8	-	-	-	(238, 543, 910)	40
A4	-	-	2	-	-	5	2	4	-	2	3	-	-	(1181, 2369, 3957)	45
A5	-	1	-	1	-	-	1	-	-	4	3	-	-	(299, 592, 962)	30
A6	-	-	-	-	-	-	-	-	-	11	3	1	-	(45, 74, 93)	26
A7	-	2	-	1	8	2	-	-	-	12	-	-	-	(964, 2077, 3450)	45
A8	3	-	-	3	1	-	6	3	-	18	5	-	-	(697, 1454, 2472)	45

Tables 4.6 and 4.7 present the classification of the software processes (i.e. functional components) of the case study used in this experiment (i.e. the uObserve software system) and the functional size approximation calculated by each participant using the *E&Q* COSMIC table. These tables also present the effort expended in minutes by each participant in conducting the experiment.

Table 4.7 Classifications of the participants with an average (1) year of experience

Participant Code	Macro Process			General Process			Typical Process			Functional Process				Functional size (min, most-likely, max)	Effort (minutes)
	S	*M*	*L*	*S*	*M*	*L*	*S*	*M*	*L*	*S*	*M*	*L*	*V. L.*		
A9	-	-	-	4	2	-	2	1	-	-	3	2	-	(250, 545, 909)	32
A10	-	9	-	-	1	1	-	-	-	-	1	-	-	(2265, 4510, 7408)	60
A11	1	1	-	-	2	1	-	-	1	3	5	1	-	(581, 1185, 1972)	23
A12	-	-	-	1	-	-	1	-	-	5	2	-	-	(57, 114, 179)	40

4.4.2 Evaluation of the reproducibility of the functional size approximation

To evaluate the reproducibility of the *E&Q* COSMIC technique, the approximations of the functional size of the 12 participants are compared to the median functional size approximation. For this data set, the median is represented by approximation of participant A2 (see Table 4.8). Therefore, the percentage difference in functional size approximation for participant A2 is (Min: 0%, Most-likely: 0%, Max: 0%), and the average percentage difference in approximation is calculated using the percentage difference of the other 11 participants. The plus sign in Table 4.8 indicates an increase in the percentage difference of the functional size approximation, and the minus sign in Table 4.8 indicates a decrease.

Table 4.8 Percentage difference in functional size approximation

Participant code	Approximate functional size using the *E&Q* COSMIC technique (Min, Most-likely, Max) (in CFP)	Percentage difference in functional size approximation (Min, Most-likely, Max)
A6	(45, 74, 93)	(−82%, −90%, −93%)
A12	(57, 114, 179)	(−77%, −84%, −86%)
A3	(238, 543, 910)	(−5%, −23%, −27%)
A9	(250, 545, 909)	(0%, −23%, −27%)
A5	(299, 592, 962)	(+20%, −16%, −23%)
A2	(250, 705, 1250)	(0%, 0%, 0%)
A1	(521, 1071, 1616)	(+108% , +52%, +29%)
A11	(581, 1185, 1972)	(+132% , +68%, +58%)
A8	(697, 1454, 2472)	(+179%, +106%, +98%)
A7	(964, 2077, 3450)	(+286% , +195%, +176%)

Table 4.8 Percentage difference in functional size approximation (Continued)

Participant code	**Approximate functional size using the *E&Q* COSMIC technique (Min, Most-likely, Max) (in CFP)**	**Percentage difference in functional size approximation (Min, Most-likely, Max)**
A4	(1181, 2369, 3957)	(+372% , +236%, +217%)
A10	(2265, 4510, 7408)	(+806% , +540%, +493%)
Minimum		**(−82%, −90%, −93%)**
Maximum		**(+806%, +540%, +493%)**
Average percentage difference relative to the functional size approximation of participant A2 (for all 12 participants)		**(+158%, +87.4%, +74%)**
Average percentage difference in the functional size approximations of participants A3, A9, A5		**(+5%, −20.6%, −26%)**

Of the twelve 12 participants in the experiment, the functional size approximations calculated by participants A3, A9, and A5 look like 'reproducible' approximations relative to the median, which is represented by the approximation of participant A2. Even though the functional size approximation of participants A3, A9, and A5 look like 'reproducible' approximations, their approximations of the functional size yield the following average percentage difference of (Min: +5%, Most-likely: −20%, Max: −26%).

Overall, the average percentage difference for the 12 participants is (Min: +158%, Most-likely: +87.4%, Max: +74%), which indicates non-reproducible results for most of the participants. The sources of large variations in the approximations of the functional size presented in Table 4.8 which yield an average percentage difference of (Min: +158%, Most-likely: +87.4%, Max: +74%) are the incorrect identification of the number of software processes and the incorrect classification made for the software processes (i.e. the functional components) in the case study. Overall, the results presented in table 4.8 indicate that the use of the rules and concepts of the *E&Q* COSMIC technique by the 12 participants does not provide a 'reproducible' approximation of the functional size of the case study used in the experiment.

4.4.3 Evaluation of the accuracy of the functional size approximation

The functional size approximations of the 12 participants in Tables 4.6 and 4.7 are first compared with the average functional size of 79.3 CFP (see Table 4.4) which was measured by the team of experts using the original version of the SRS document. The Magnitude of Relative Error (MRE) equation is used to calculate the accuracy of the functional size approximations (see Table 4.9) as follows:

- the 1st column presents the functional size values approximated by the 12 participants in the experiment;
- the 2nd column presents the average functional size value measured by the team of experts (considered here as the reference value for accuracy); and
- the 3rd column presents the MREs calculated using the approximate functional sizes from the 1st column and the average measured functional size from the 2nd column.

Table 4.9 Accuracy of the functional size approximation - 12 participants

Participant code	**Approximated functional size using the *E&Q* COSMIC technique in CFP (min, most-likely, max) (1)**	**Reference functional size for accuracy criteria (2)**	**MRE calculated using values in (1) and (2) (min, most-likely, max)**
A6	(45, 74, 93)	79.3 CFP	(43%, 7%, 17%)
A12	(57, 114, 179)		(28%, 44%, 126%)
A3	(238, 543, 910)		(200%, 585%, 1047%)
A9	(250, 545, 909)		(215%, 587%, 1046%)
A5	(299, 592, 962)		(277%, 646%, 1113%)
A2	(250, 705, 1250)		(215%, 789%, 1476%)
A1	(521, 1071, 1616)		(557%, 1251%, 1938%)
A11	(581, 1185, 1972)		(633%, 1394%, 2387%)
A8	(697, 1454, 2472)		(779%, 1733%, 3017%)
A7	(964, 2077, 3450)		(1115%, 2519%, 4250%)
A4	(1181, 2369, 3957)		(1389%, 2887%, 4890%)
A10	(2265, 4510, 7408)		(2756%, 5587%, 9241%)
Average MRE on functional size approximations (all 12 participants)			**(684%, 1502%, 2546%)**
Average MRE on functional size approximations (except participants A6 & A12)			**(814%, 1798%, 3041%)**

Of the functional size approximations calculated by the 12 participants in the experiment, only those calculated by participants A6 and A12 look like 'reasonable' approximations relative to the reference average functional size of 79.3 CFP:

- the functional size approximations of A6 and A12 resulted in 'most-likely' MRE values of 7% and 44%, respectively, while the MREs of the remaining 10 participants vary wildly from +500% to +5000%;
- for these 10 participants, the average MREs of the functional size approximations are: Min: 814%, Most-likely: 1798%, Max: 3041% (bottom line of Table 4.9).

Overall, for the latter 10 participants, the functional size approximation range of MRE values is (Min: 814%, Most-likely: 1798%, Max: 3041%). Overall, the average MRE for the 12 participants is (Min: 684%, Most-likely: 1502%, Max: 2546%), which indicates extremely highly inaccurate results for most of the participants.

The source of these inaccurate results is the high level of misclassification of the software processes by the participants. For instance:

- participant A1 identified data movements in two software processes as a set of 18 ‘very large’ functional processes and 8 ‘small’ General Processes. This large number of software processes identified by this participant leads to a range of size approximations (Min: 521 CFP, Most-likely: 1071 CFP, Max: 1616 CFP) using the statistical table of the *E&Q* COSMIC technique (i.e. Table 4.3). This participant (A1) had 13 years of experience in requirements analysis and modeling at the time of the experiment.
- participants A2 to A8 classified most of the software processes, if not all, at higher levels of classification, and neglected all the available functional specifications of the software processes (i.e. the functional components). Participant A4 had 8 years of experience in requirements analysis and modeling at the time of the experiment, but he classified 13 software processes at the Macro, General and Typical processes levels, and only 5 software processes as functional processes. Participant A4 identified 2 large macro processes as well as five (5) large general processes, leading to a very large range of

approximations (Min: 1181 CFP, Most-likely: 2369 CFP, Max: 3957 CFP) using the table of the *E&Q* COSMIC technique.

- participant A10 identified twelve (12) software processes in the case study, and failed to classify them in their correct class of the *E&Q* COSMIC functional components, in accordance to their level of granularity. This led to a very large range of approximation sizes: (Min: 2265 CFP, Most-likely: 4510 CFP, Max: 3957 CFP) using the table of the *E&Q* COSMIC technique.

Tables 4.10 and 4.11 present the number of software processes identified by each of the 12 participants in the experiment. The Magnitude of Relative Error (MRE) equation was used to calculate the accuracy of the number of software processes identified. That identified number of software processes was then compared with the reference value of 15 software processes prepared by the designer of this experiment and verified for correctness by the independent expert. Tables 4.10 and 4.11 present:

- the number of software processes identified by the 12 participants (1st column);
- the correct number of software processes identified by the principal researcher and verified by the independent expert (2nd column); and
- the corresponding Magnitude of Relative Error (MRE) values (3rd column) calculated using the values from the 1st and 2nd columns.

Table 4.10 Number of software processes identified by participants A1 to A8

Participant code	**Number of software processes identified (1)**	**Reference no. of software processes (2)**	**MRE using values from (1) & (2)**
A1	55	15	266%
A2	10		33%
A3	18		20%
A4	18		20%
A5	10		33%
A6	15		0%
A7	25		67%
A8	39		160%
Average MRE on software processes			**74%**

Table 4.11 Number of software processes identified by participants A9 to A12

Participant Code	Number of software processes identified (1)	Reference no. of software processes (2)	MRE using values from (1) & (2)
A9	14	15	7%
A10	12		20%
A11	15		0%
A12	9		40%
Average MRE on software processes			**17%**

Only participant A6 in table 4.10 was able to identify the correct number of software processes explained in the case study. However, this participant could not classify them in accordance to their levels of granularity in the correct *E&Q* functional classes – see table 4.6.

In addition, only participant A11 in Table 4.11 was able to identify the correct number of software processes explained in the case study, but he misclassified them, which led to a large range of size approximations (Min: 581 CFP, Most-likely: 1185 CFP, Max: 1972 CFP). Furthermore, the functional size approximations of participant A12 look 'reasonable' (see table 4.7). However, participant A12 identified only 9 software processes, instead of the correct number of 15 software processes and could not classify them in accordance to their levels of granularity in the correct *E&Q* functional classes.

It is worth mentioning that the average MRE of 17% of the participants in Table 4.11 (i.e. participants with limited industry experience) is great deal better (i.e. a smaller MRE) than the average MRE of 74% for the participants in Table 4.10 (i.e. participants with 12 years of industry experience). This is because the participants with limited industry experience identified less software processes than the participants with 12 years of industry experience.

Most of the participants in Tables 4.6 and 4.7 calculated inaccurate functional size approximations, since they had incorrectly identified and classified the software processes (i.e. the functional components) of the case study. Overall, the results presented in this subsection indicate that use of the rules and concepts of the *E&Q* COSMIC technique by the

12 participants did not help them arrive at an 'accurate' approximation of the functional size of the case study used in the experiment.

4.4.4 Summary of findings

The experimental results presented in sections 4.4.2 and 4.4.3 lead to the following findings:

- the functional size approximations calculated by the 12 participants using only the rules and the concepts of the *E&Q* COSMIC technique did not lead to reproducible or accurate results in this experiment.
- the incorrect identification and classification of the functional components had in impact on the reproducibility and accuracy of the functional size approximations of the functional components.
- the participants with extensive industry experience and those with limited industry experience made similar mistakes, in terms of incorrectly identifying and classifying of the software processes in the case study. In other words, the participants with extensive industry experience did not perform better than those with limited industry experience.

4.5 Validity Threats

4.5.1 Construct validity threats

A construct validity threat is associated to the failure of the experimental setting to reflect the conditions of the technique under study (i.e. the *E&Q* COSMIC technique). In the case of the experiment reported here, the type of reference manual mentioned in (DPO, 2007) for the *E&Q* COSMIC technique was not available. To mitigate the risk of this type of threat occurring, the experimental material that was made available to the participants in the experiment was designed to contain equivalent information to that in the reference manual of the *E&Q* FPA technique (DPO, 2007). In other words, the material used in the experiment contains a complete description of the *E&Q* COSMIC technique, including the functional size approximation rules and procedures.

The participants in the experiment were not able to correctly classify the software processes in accordance with their levels granularity, as described ‘as is’ in the proposed *E&Q* COSMIC technique. The preliminary steps recommended in (COSMIC, 2007) were not taken into account in the experiment design, owing to the unavailability of related guidelines from the literature fifteen years after the initial publication of the *E&Q* technique in (Meli, 1997) and eight years after the publication of the COSMIC variant in (Conte, Iorio et Santillo, 2004).

A second construct validity threat is the restricted time available in which to conduct the experiment. This lack of time prevented the participants from asking for clarification from their colleagues, or from experts in the field, which is common practice in software development organizations.

4.5.2 Internal validity threats

An internal validity threat is associated with any changes in the design of the experiment, such as lack of discussion or clarification during the experimental period, lack of clear data collection procedures, or description of the concept(s) to be evaluated in the experiment, that could affect the validity of the experimental results. To mitigate the risk of this type of validity threat occurring, a one-hour tutorial session was held prior to the experiment to describe its objectives, scope, and rules, as well as the roles of the participants. The principal researcher explained the *E&Q* COSMIC technique in detail, and opened the door to discussion to clarify the activities and materials of the experiment, including a complete description of the *E&Q* COSMIC technique, a participant experience survey, and data collection forms.

Moreover, the principal researcher conducted a pilot test of the experiment by performing the experimental activities of the prior to running the actual experiment. This was done with the help of an independent expert, in order to identify any potential challenge in the experimental procedures, including the applicability of the SRS to the experiment, the time required to

conduct the experiment, and the usability of the data collection forms to be used by the participants in the experiment, as well as to verify the correctness of the reference classification of the functional components of the software system. The independent expert had 20 years of experience in requirements analysis and modeling, 6 years of experience in software documentation and software quality assurance, and 3 years of experience in functional size measurement using the COSMIC measurement method.

The independent expert identified the correct number of software processes (15) explained in the requirements document in (Trudel et Lavoie, 2008), and classified 13 of them in the reference classification proposed by the principal researcher. However, the independent expert classified 1 of the software processes as a large functional process, whereas this functional process was deemed by the principal researcher to be a medium functional process. The independent expert identified 3 more data movements than the principal researcher, and this affected the total number of data movements identified in that functional process and resulted in its classification as a large functional process.

Similarly, the independent expert classified another software process as a medium functional process, whereas this functional process was deemed by the principal researcher to be a large one. The independent expert identified 2 data movements fewer than the principal researcher, and this affected the total number of data movements identified in that functional process and its classification.

The differences in the classification of the 2 software processes were caused by assumptions made by the independent expert for elements in the Graphical User Interface (GUI) of the software system. This affected the identification of the data movements in each software process. These differences should not be considered as misclassifications, because they reflect the various ‘flavors’ of functional behavior – as a result of the assumptions – of the software (Trudel, 2012). Also, the differences in the classification of these software processes did not affect the final functional size approximation of the software.

Next, the Magnitude of Relative Error (MRE) equation was used to calculate the accuracy of the functional size approximation in Table 4.5 relative to the reference average functional size of 79.3 CFP (see Table 4.4). This gives a range of MRE values of (Min: 28%, Most-likely: 8.4%, Max: 36.2%). It is worth noting that the approximate 'Most-likely' functional size of 87 CFP is close to the reference average functional size of 79.3 CFP (i.e. it yields an MRE value of 8.4%).

The experiment was designed to apply the *E&Q* COSMIC technique using a single case study (i.e. uObserve requirements specifications) with a group of 12 participants. In other words, the experiment tested the reproducibility of the classification process with multiple subjects (i.e. participants) using the same requirements document, in order to obtain multiple ranges of functional size on the same requirements document. Assessment of the ranges introduced in the analytical table was outside the scope of the design of this experiment.

Another potential threat is that all the software processes described in the case study document were only functional processes, and none were higher-level processes (Macro, General, or Typical). To mitigate this threat, future experiments will be designed to apply the *E&Q* COSMIC technique using multiple case studies with higher process levels (i.e. Macro, Generic, Typical) in order to assess the ranges introduced in the analytical/statistical table (i.e. minimum, most likely, and maximum size values).

4.5.3 External validity threats

One external validity threat here is associated with the failure to be able to generalize the experimental results beyond the experimental setting. The number of participants in the experiment was limited to 12. However, the experiment involved participants with 2 profiles: experienced participants, and participants with limited experience. Participants A1 to A8 had significant experience in software requirements analysis, modeling, and quality assurance, while participants A9 to A12 had limited experience in these areas. In spite of this, the

classification results showed that they all committed similar errors in classifying the software processes of the software system.

4.6 Chapter Summary

This chapter looked into the application of the *Early & Quick* COSMIC technique using a single case study (i.e. uObserve requirements specifications). The experiment tested the reproducibility and accuracy of the functional size approximations with multiple subjects (i.e. participants) using the same requirements document. The functional size approximations produced by 12 participants from the software engineering industry using only the rules and concepts of the *E&Q* COSMIC technique currently available to the industry did not lead to results that were either reproducible or accurate:

- The average MRE of the functional size approximation of the 12 participants relative to the reference average functional size of 79.3 CFP is as follows: Min MRE 684%, Most likely MRE 1502%, Max MRE 2546%.
- The average percentage difference in functional size approximation relative to the median approximation is (Min: +158%, Most-likely: +87.4%, Max: +74%).
- Only 2 participants were able to identify the correct number of software processes: the average MRE of the number of identified software processes of participants with 12 years of industry experience is 74%, and the average MRE of the number of identified software processes of participants with limited industry experience is 17%.
- None of the 12 participants in the experiment classified the identified software processes in the correct *E&Q* functional classes, in accordance with their levels of granularity.

This experiment could not take into consideration the two preliminary steps recommended in (COSMIC, 2007) for the application of the *Early & Quick* COSMIC technique:

- identification of the levels of granularity of the software requirements specifications; and
- identification and use of size scaling factors.

This was mainly because of the unavailability in the literature of guidelines for performing these two steps, even though 15 years has passed since the initial publication of the *Early & Quick* techniques and 8 years has passed since the publication of the COSMIC variant (Conte, Iorio et Santillo, 2004). The implicit assumption in (COSMIC, 2007) is that such guidelines would lead to reasonably accurate and reproducible approximations, but there is no supporting evidence that this assumption works as intended.

The industry and the research community recognize the importance of approximate sizing, and size approximation techniques, like the *E&Q* COSMIC technique, have been proposed, which consist of:

a) a procedural part (i.e. identification and classification of the functional components of software); and
b) assignment of the numerical size values of the classified functional components using tables of size factors, such as the *E&Q* COSMIC statistical table.

This experiment has used all information available on such a technique, but could not demonstrate that it led to either reproducible or accurate results. All the available information on such a technique has been used in this experiment, but we could not demonstrate that it led to either reproducible or accurate results.

The software measurement industry has recognized that guidelines are needed, but none has been put into the public domain. Consequently, it has yet to be demonstrated that guidelines lead to reasonably reproducible and accurate results. In summary, there is no documented evidence that:

a) the initial *E&Q* COSMIC design works as intended; or that
b) the preparatory guidelines designed to support these approximation techniques work as intended.

The software measurement industry and the researchers need to work on developing such guidelines and on verifying that they work as intended. It must be shown that they lead to:

- a reproducible approximation of functional size; and

- a reasonably accurate approximation of functional size.

The methodology used in this experiment can be reused to test the contributions of guidelines as they become available in the public domain.

In addition, the case study used in this experiment and the quantitative findings of this research can be used as a benchmark to quantitatively test the contributions of any guidelines that are proposed in the future by researchers or practitioners.

The experiment reported here is part of a research project aimed at designing a framework to identify the levels of granularity of software requirements specifications and to assign scaling factors to them to rank their levels of granularity. In other words, the research objective is to design a framework that takes into account the two preliminary steps recommended in (COSMIC, 2007), on which no details were provided by the authors of (Meli, 1997), (Conte, Iorio et Santillo, 2004) on how to apply these steps in practice.

CHAPTER 5

DESIGN OF THE SCALING FACTORS FRAMEWORK

5.1 Introduction

This chapter presents the design of the scaling factors framework aimed to help the requirements engineers to identify the levels of granularity of functional requirements specifications of software development projects. It includes identifying the relevant concepts needed for the specification of the functional requirements of software and for defining the format and the type of the scaling factors. This chapter also includes designing a meta-model to capture the identified relevant concepts with the defined scaling factors. Further, this chapter includes the design of criteria to identify the levels of granularity of the functional requirements of software development projects.

The main objective of this research project is to design a framework that assign scaling factors for identifying the levels of granularity of functional requirements specifications of software, which are typically documented at different levels of granularity at early stages of the software development life cycle.

The steps to design the framework are given as follows:

1) **Meta-model design:** the purpose of the meta-model is to define the relevant concepts that will form the basis for the design of the set of criteria: this meta-model is to capture the relevant concepts that allow for full-specification of functional requirements specifications of software. The meta-model will be designed in two variants:
   - **A)** 1st variant of the meta-model will capture in a generic manner the relevant functional components of a software application and without specifying how these functional components are documented.
   - **B)** 2nd variant of the meta-model will capture in a specific manner the relevant functional components of a software application by using elements from the COSMIC

measurement method and elements of the use-case model (i.e. UML use-cases and use-case scenarios).

2) **Criteria:** a set of criteria is designed to help the requirements engineers in identifying the levels of granularity of the functional requirements specifications. The set of criteria is designed in the same manner of the meta-model design:
   1) the set of criteria identifies the levels of granularity of the software functional components and without specifying how the functional components are documented.
   2) the set of criteria maps the generic software functional components to the elements of the COSMIC measurement method and elements from the UML use-case model.

3) **Interval scaling factors:** this step is aimed to convert the set of ordinal scaling factors proposed in Step (1) into a set of interval scaling factors in order to rank the level of granularity of functional requirements specifications using the measurement unit of the international standard for software functional size measurement COSMIC - ISO19761. More details are presented in Chapter 7.

In this research project, we take as input the Functional User Requirements (FURs) of software applications. Figure 5.1 presents a generic representation of a software application along with its functional components and sub-components. The software application presented in figure 5.1 consists of N layers (i.e. layer architecture), where each layer consists of several software components. The software components in each layer exchange messages between each other and with other software components in other layers. It is worth mentioning that for the purpose of this research project, we consider each software layer as a separate software that exchange messages with the external environment (i.e. other layers & functional users outside the software boundary) through its boundary.

This chapter is organized as follows: Section 5.2 presents the definition of the elements of the meta-model and the design of the meta-model structure. Section 5.3 presents the design of the set of criteria aimed to identify the levels of granularity of the software requirements specifications. A summary of this chapter is presented in Section 5.4.

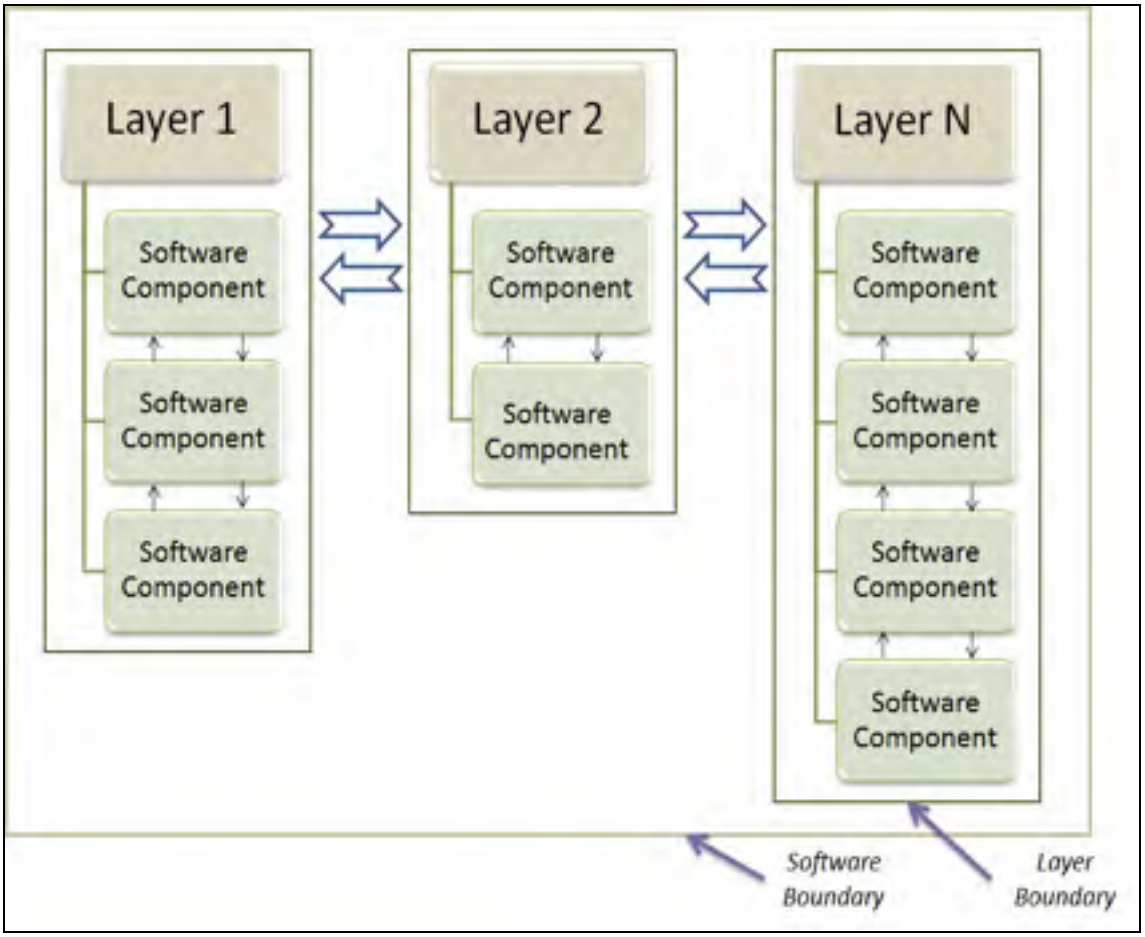

Figure 5.1 Generic representation of a software application

5.2 Meta-Model Design

Before designing the set of criteria that assign scaling factors that identify the level of granularity of early versions of functional requirements specifications, it is necessary to define the relevant conceptual constructs that will form the basis for the design of the set of criteria: such conceptual constructs will allow the requirements engineers to document full-functional specification of functional requirements specifications of software.

The Extended Meta Object Facility (EMOF) specification in (Eclipse, 2010) is selected as a means of designing the meta-model, for the following reasons:

- it allows the definition of the abstract syntax of a modeling phenomenon;
- it allows the structural definition of the involved conceptual constructs with their properties, the definition of relationships among the different constructs; and
- it allows the definition of a set of rules to control the interaction among the different conceptual constructs.

As part of the Extended Meta Object Facility (EMOF), a meta-model in (Eclipse, 2010) helps in capturing the relevant conceptual constructs (i.e. software requirements specifications) along with their relationships: such capturing of these conceptual constructs and their relationships helps the personnel involved in the functional size measurement process to build the necessary software model to be used as an input for the functional size approximation process. The UML class diagram (Arlow et Neustadt, 2005) is used to represent the meta-model for its industry usage, simplicity, and expressiveness.

The meta-model is designed in two (2) variants:
The 1st variant meta-model (see Figure 5.2) is designed to capture in a generic manner the relevant functional components of a software application and without specifying how those functional components are documented.

As can be observed in the meta-model presented in Figure 5.2:

- a *piece of software* can be partitioned into multiple *software layers*, in which each *software layer* is separated from other layers by a layer *boundary*;
- a *software layer* may consist of several *software components*; and
- these *software components* may consist of one or more *software sub-components*.

The *software sub-components* can be classified as follows:

- **<implicit> sub-component:** the implicit sub-component – that is specialized from the software sub-component class – is only inferred:
  - not explicitly mentioned in the functional user requirements; and

- there exists no detail to precisely identify it and there exist no detail that describe its functional specification.

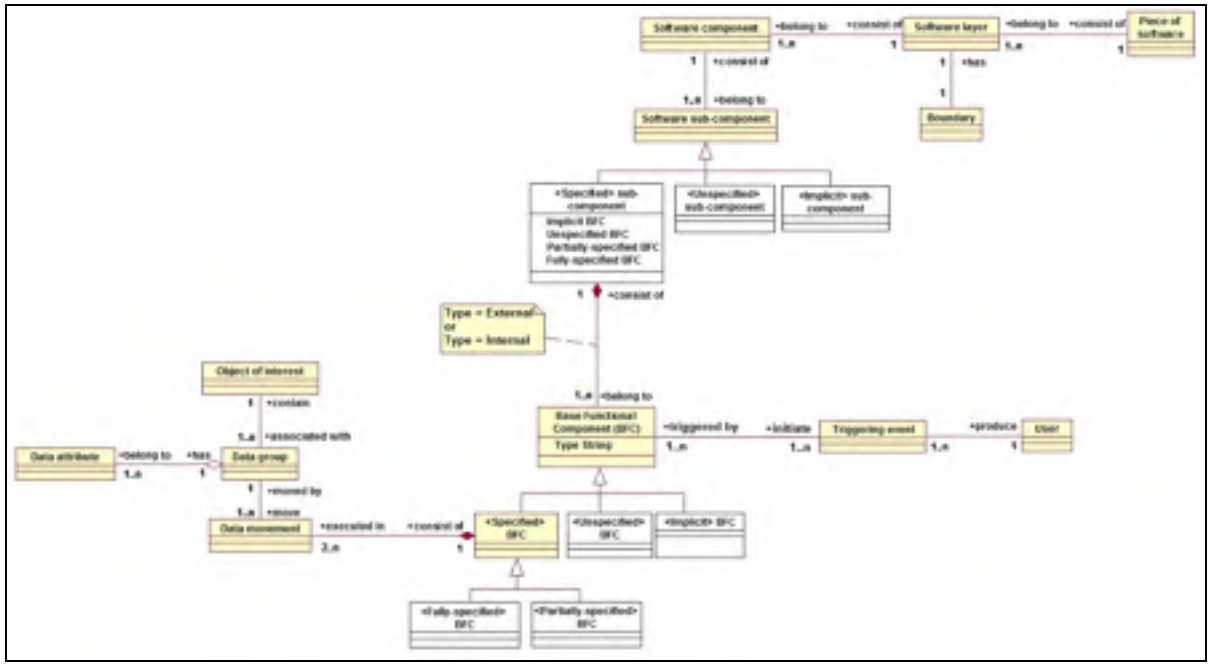

Figure 5.2 1st variant of the meta-model that captures the relevant concepts of a software application along with their relationships

- **<unspecified> sub-component:** the unspecified sub-component is precisely identified:
  - explicitly mentioned in the functional user requirements, and there exist details to precisely identify it; but
  - there exists no detail that describes its functional specification.

- **<specified> sub-component:** the specified sub-component is precisely identified:
  - explicitly mentioned in the functional user requirements, and there exist details that describe its functional specification, including:
    - the triggering event(s); and
    - the functional user(s) who initiates the communication with the software application.

The <*specified*> *sub-component* can be further decomposed into four (4) possible types of Base Functional Components (BFCs):

1) 'implicit';
2) 'unspecified';
3) 'partially-specified'; and
4) 'fully-specified'.

For this reason, the <specified> sub-component class presented in figure 5.2 contains four (4) attributes: each attribute indicates the number of BFCs of a specific type exists in that <specified> sub-component. For instance, if the attribute <Implicit> BFC = 3; this means that there exist three (3) 'implicit' BFCs in that <specified> sub-component.

A Base Functional Component (BFC) is generally defined as follows: *"It is an elementary unit of the functional user requirements defined by a functional size measurement method for measurement purposes"* (ISO19761, 2011).

The BFC is triggered by at least one triggering event that is produced by a user of the software application. This user can be the functional user who is interacting with the software through its boundary, or another BFC in the same/different software (sub) component. In the first case, the BFC is assigned a 'Type' attribute value 'External'), and in the latter case, the 'Type' attribute is then assigned a value 'Internal' to indicate that the BFC is triggered from another BFC and not from the functional user of the software application.

A *data group* is any distinct, non-empty, non-ordered and non-redundant set of data attribute types where each included data attribute type describes a complementary aspect of the same object of interest (ISO19761, 2011). A data attribute type is the smallest parcel of information, within an identified data group type, carrying a meaning from the perspective of the software FURs (ISO19761, 2011), and Object of interest type is any "thing" that is identified from the point of view of the FURs. It may be any physical thing, as well as any

conceptual object or part of a conceptual object in the world of the user about which the software is required to process and/or store data.

As described above, a <Specified> sub-component can be decomposed into four (4) possible BFCs; these BFCs can be classified as follows:

- **<implicit> BFC:** the implicit BFC is only inferred:
  - not explicitly mentioned in the functional user requirements, and there exist no details to precisely identify it; and
  - no detail exists to describe its functional specification.

- **<unspecified> BFC:** the unspecified BFC is precisely identified:
  - explicitly mentioned in the functional user requirements; but
  - there exists no detail to describe its functional specification.
- **<specified> BFC:** the specified BFC is precisely identified:
  - explicitly mentioned in the functional user requirements, and there exist details to precisely identify it, and
  - there exist details that describe its functional specification.

The <*specified*> BFC can be either a <*Partially-specified*>BFC or <*Fully-specified*> BFC and they are defined as follows:

- **<partially-specified> BFC:** is partially documented: i.e. the data movements are not precisely identified and measured along with their associated data groups.
- **<fully-specified> BFC:** is completely documented: i.e. the data movements are precisely identified and measured along with their associated data groups.

Instances of <*Partially-specified*> BFC or <*Fully-Specified*> BFC must contain at least two data movements, which are:

- clearly specified by the cardinality of rank ‘2’ next to the data movement class, and
- “filled diamond” notation to indicate that any instance of <*Partially-specified*> BFC or <*Fully-specified*> BFC classes must have at least two data movements.

The 2nd variant of the meta-model in figure 5.3 is aimed to specify a mapping between the generic software (sub) components presented in figure 5.2 and elements from the COSMIC measurement method and UML use-case model. A software sub-component corresponds to a use-case in the UML use-case model, and a Base Functional Component (BFC) corresponds to a use-case scenario - a Functional Process in COSMIC - in the UML use-case specification. For the purpose of this research project, we adopt the definition of the Functional Process type in the COSMIC measurement method (ISO19761, 2011) as our definition for the Base Functional Component, as follows:

> *"An elementary component of a set of Functional User Requirements comprising a unique cohesive and independently executable set of data movement types: It is triggered by a data movement from a functional user that informs the piece of software that the functional user has identified a triggering event. It is complete when it has executed all that is required to be done in response to the triggering event type"* (ISO19761, 2011).

A data movement can be one of the following four types:

- Entry: is a data movement type that moves a single data group from a functional user across the software boundary into the BFC;
- eXit: is a data movement that moves a single data group from a BFC across the boundary to the functional user;
- Read: is a data movement that moves a single data group from persistent storage within reach of the BFC which requires it; and
- Write: is a data movement that moves a single data group lying inside a BFC to persistent storage.

The mapping from the generic software (sub) components into the corresponding elements in the COSMIC measurement method and UML use-case model as follows:

- **<implicit> use-case:** the implicit use-case is only inferred:
  - not explicitly mentioned in the functional user requirements, there exist no details to precisely identify it; and

- there exist no detail that describes its functional specification.

- **<unspecified> use-case:** the unspecified use-case is precisely identified:
  - explicitly mentioned in the functional user requirements, and there exist details to precisely identify it, but
  - there exists no detail that describes its functional specification.

- **<specified> use-case:** the specified use-case is precisely identified:
  - explicitly mentioned in the functional user requirements; and
  - there exist details that describe its functional specification, including:
    - triggering event(s); and
    - the functional user(s) who initiate the communication with the software.

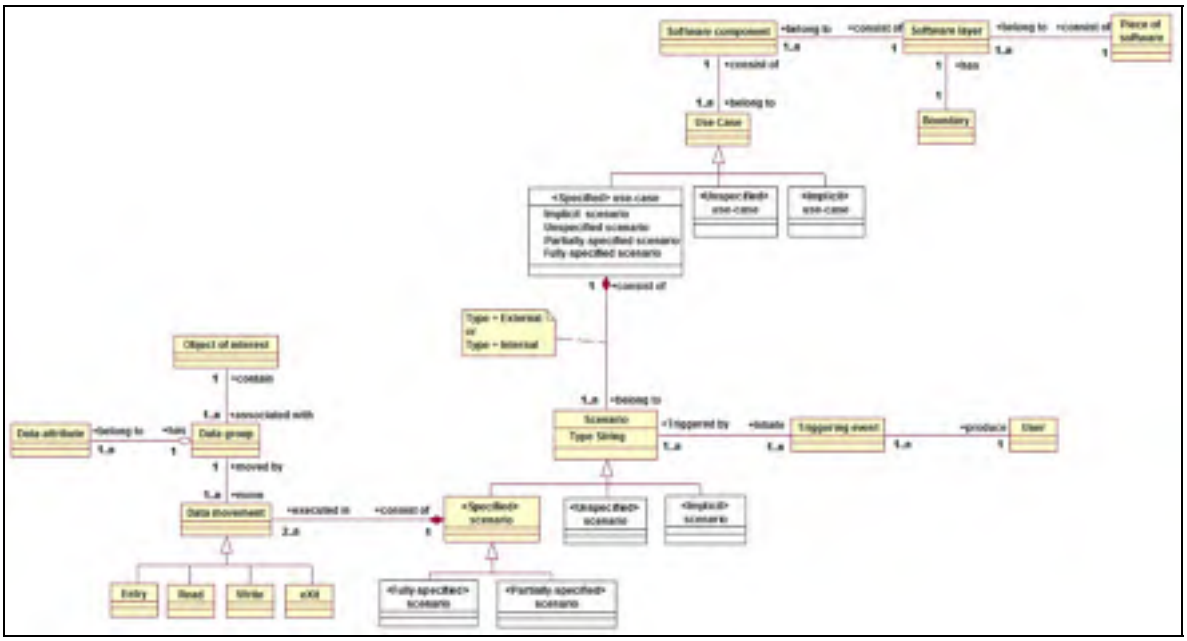

Figure 5.3 2nd variant of the meta-model that captures the corresponding elements in the UML use-case model

The <*specified*> *use-case* can be further decomposed into four (4) possible types of use-case scenarios:

- 'implicit';
- 'unspecified';

- ‘partially-specified’; and
- ‘fully-specified’.

For this reason, the <*specified*> *use-case* class contains four (4) attributes: each attribute indicate the number of use-case scenarios of a specific type exist in the functional specification of that <*specified*> *use-case*. For instance, if the attribute <*Implicit*> *scenario* = 2; this means that there exist three (2) ‘implicit’ scenarios in that <*specified*> *use-case*.

- **<implicit> scenario:** the implicit use-case scenario is only inferred:
  - not explicitly mentioned in the functional user requirements, there exist no details to precisely identify it; and
  - no detail that describes its functional specification.

- **<unspecified> scenario:** the unspecified use-case scenario is precisely identified:
  - explicitly mentioned in the functional user requirements; but
  - there exists no detail that describes its functional specification.

Finally, the **<Specified> scenario** can be either a <*Partially-specified*> or <*Fully-specified*>*scenario*: it can be precisely identified and there exist details that describe its functional specification:

- **<partially-specified> scenario:** partially documented: i.e. the data movements are not precisely identified and measured along with their associated data groups; and
- **<fully-specified> scenario:** completely documented: i.e. the data movements are precisely identified and measured along with their associated data groups.

It is worth mentioning that the <*Specified*> *use-case scenario* class has a *‘Type’* attribute: this attribute has two values *‘External’* and *‘Internal’*:

- if the <*Specified*> *use-case scenario* is triggered by the functional user that is interacting with the software through its boundary, then it is assigned an attribute value *‘External’*.

- if it is triggered by another use-case scenario in the same or different software use-case, it is assigned a type attribute value *'Internal'*.

An example for an 'External' BFC can be the 'Login Main flow' scenario in a Login use-case, in which the functional user of the software application interacts with the software through its boundary. Whereas, an example for an 'Internal' BFC can be 'Invalid user credentials' scenario in the Login use-case: this use-case scenario is triggered when the functional user of the software application enters incorrect login information.

5.3 Criteria

The set of criteria is designed to identify the level of granularity of each identified software component and its associated software sub-components (use-cases), including Base Functional Components (scenarios) in order to assign them scaling categories identified in Section 5.2 to rank their level of granularity.

The set of criteria is designed in two variants:

1. 1st variant of the set of criteria, which is designed to identify the level of granularity of software functional components and without specifying how those functional components are documented (see Figure 5.4).
2. 2nd variant in the set of criteria identifies the level of granularity of the software functional components, where the software functional components are represented by elements from COSMIC measurement method and the UML use-case model (i.e. use-cases and use-case scenarios) (see Figure 5.5).

It is worth mentioning that the process of decomposing the FURs into separate components is out of the scope of this research project. However, this process should be conducted when the Generic Criterion (GC1) is not met and before proceeding to the remaining set of criteria.
The Generic Criterion (GC2) checks for each component of the software application: it checks whether the sub-components (if available) are precisely identified and counted:

- if the sub-components of a specific software component meet this criterion, they are passed on to the next criterion.
- otherwise the sub-components are only inferred (i.e. not explicitly mentioned in the functional user requirements, and there exist not enough details to precisely identify it), and no detail exists to describe its functional specification.

Therefore, each software sub-component is assigned a scale category *<Implicit> sub-component* to rank its level of granularity.

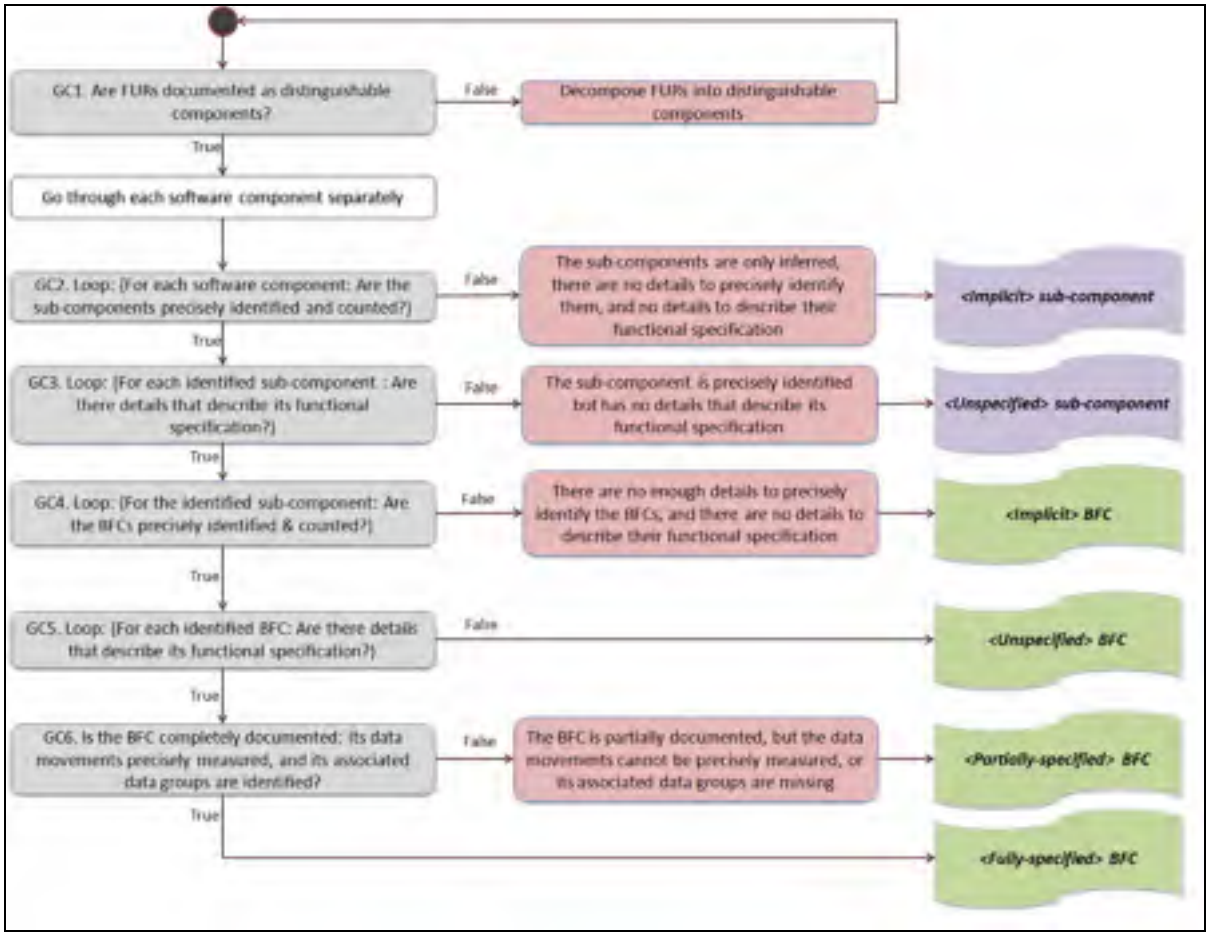

Figure 5.4 1st variant of the set of criteria to assess the levels of granularity of software functional components

Then, the Generic Criterion (GC3) checks for each software sub-component that is precisely identified to verify if there are details that describe its functional specification:

- if the sub-component is precisely identified (i.e. explicitly mentioned in the functional user requirements), but there are no detail that describes its functional specification: it is assigned the scale category <*Unspecified*> *sub-component*.
- otherwise, the sub-component is passed through the next criterion (In other words, the sub-component has details that describe its functional specification): the sub-component is assigned the scale category <*Specified*> *sub-component*, and the remaining criteria are to examine the Base Functional Components in this sub-component.

The Generic Criterion (GC4) checks for each sub-component whether there exist details to precisely identify and count the Base Functional Components (BFCs):

- if BFCs are precisely identified and counted; they are passed on to the next criterion for evaluation (i.e. Generic Criterion 5).
- otherwise, the BFCs are only inferred (i.e. not explicitly mentioned in the functional user requirements and there exist not enough details to precisely identify and measure them), and no detail exists to describe their functional specification. Then, each inferred BFC is assigned the scale category <*Implicit*> *BFC*.

The Generic Criterion (GC5) checks for each BFC precisely identified of a specific sub-component if there are details that describe its functional specification:

- if there exists no detail that describes its functional specification it is assigned the scale category <*Unspecified*> *BFC*.
- otherwise, it is passed to the next criterion for evaluation.

The Generic Criterion (GC6) checks if the BFC is completely documented:

- if it achieves the desired goal of the user of the software application, and its data movements are precisely measured, and its associated data groups are also identified: the BFC is then assigned the scale category <*Fully-specified*> *BFC*.
- otherwise, the BFC is partially documented, and the data movements are not precisely measured, or their associated data groups are missing. Therefore, the BFC is assigned the scale category <*Partially-specified*> *BFC*.

The 2nd variant in the set of criteria identifies the level of granularity of the software functional components identified in Section 5.2, which the software functional components are represented by elements from COSMIC measurement method and the UML use-case model (i.e. use-cases and use-case scenarios) - See figure 5.5: a software *sub-component* corresponds to a use-case in the UML use-case model, and a Base Functional Component (BFC) corresponds to a use-case scenario in the UML use-case specification.

The starting point in the 2nd variant of the set of criteria:

The Generic Criterion (SC1) checks whether the Functional User Requirements (FUR) are documented as separate distinguishable software components: this is accomplished by checking the Generic Criterion (SC1):

- if the FURs meet this criterion: they are passed to the Generic Criterion (SC2).
- otherwise, the FURs should be decomposed into separate distinguishable software components.

The Specified Criterion (SC2) checks for each component of the software application: it checks whether the use-cases (if available) is precisely identified and counted:

- if use-cases of a specific software component meet this criterion: they are passed to the next criterion (i.e. SC3) for evaluation.
- otherwise, the use-cases are only inferred (i.e. not explicitly mentioned in the functional user requirements, and there exist not enough details to precisely identify them, and no detail that describes their functional specification. Therefore, each use-case is assigned the scale category <*Implicit*> *use-case*.

Next, the Specified Criterion (SC3) checks for each use-case that has been precisely identified if there exist details that describe its functional specification:

- if the use-case is precisely identified (i.e. explicitly mentioned in the functional user requirements), but there is no detail that describes its functional specification: it is assigned the scale category <*Unspecified*> *use-case*.

- otherwise, the use-case is assigned a scale category <*Specified*> *use-case*, and the remaining criteria are to examine the scenarios in this use-case.

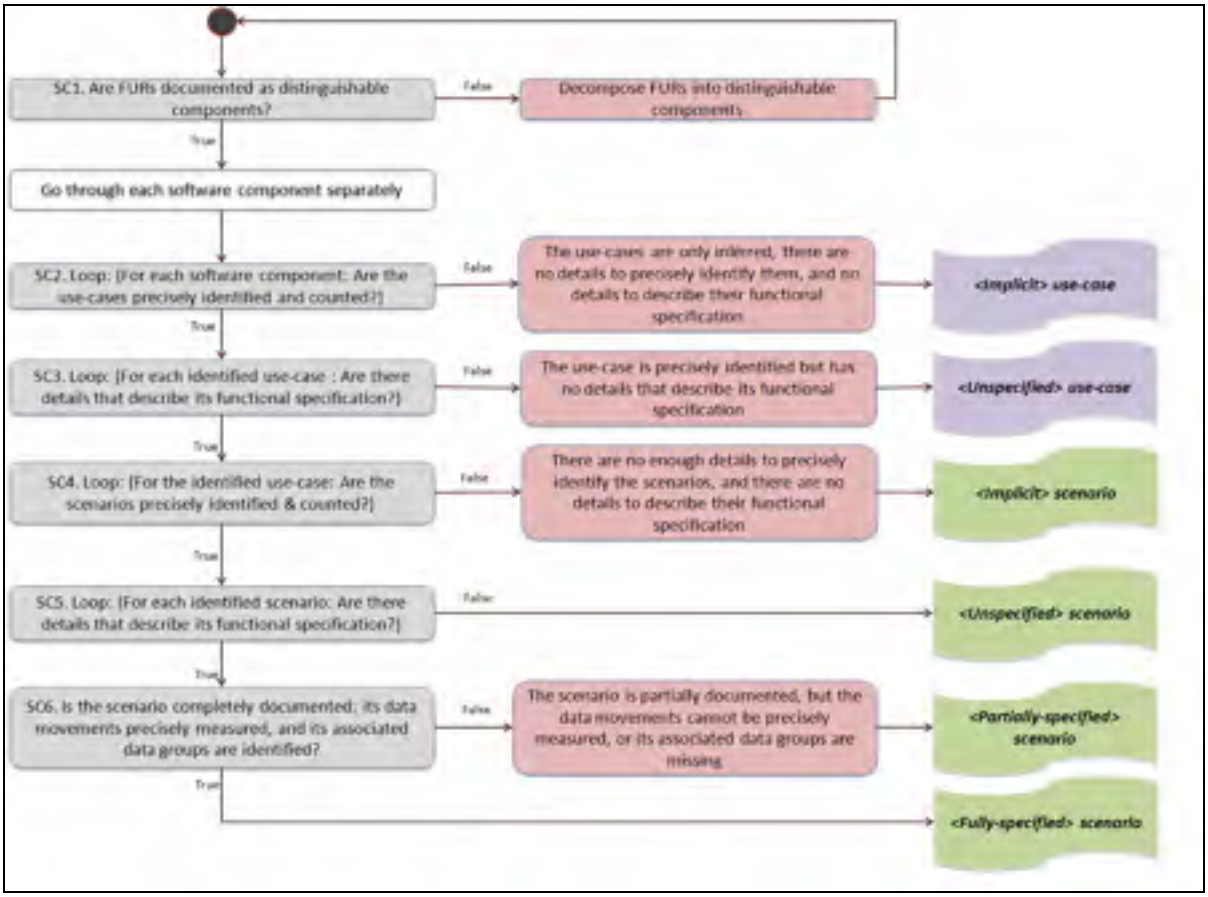

Figure 5.5 2nd variant of the set of criteria assesses the levels of granularity of the elements of the UML use-case model

The Specified Criterion (SC4) checks for each use-case: whether there are details to precisely identify and measure the scenarios in this use-case:

- if the scenarios meet this criterion: they are passed to the next one (i.e. Specified Criterion 5).
- otherwise, the scenarios are only inferred and no detail exists to describe their functional specification. Then, each scenario is assigned the scale category <*Implicit*> *scenario*.

The Specified Criterion (SC5) checks for each scenario that has been precisely identified in a <*specified*> *use-case* if there are details that describe its functional specification:

- if the scenario is precisely identified (i.e. explicitly mentioned in the functional user requirements), but there is no detail that describes its functional specification: it is assigned the scale category <*Unspecified*> *scenario*.
- otherwise, it is passed to the next criterion for evaluation.

The Specified Criterion (SC6) checks if the scenario is completely documented:

- if it achieves the desired goal of the user of the software application, and its data movements are precisely identified, and its associated data groups are also identified: the scenario is assigned the scale category <*Fully-specified*> *scenario*.
- otherwise, the scenario is partially documented, and the data movements are not precisely measured, or their associated data groups are missing. Therefore, the scenario is assigned the scale category <*Partially-specified*> *scenario*.

5.4 Chapter Summary

This chapter has presented the design of the scaling factors framework aimed to identify the level of granularity of the functional requirements specifications of software development projects. The design of the scaling factors framework has been carried out in two steps:
(1) the definition of the elements of the meta-model and then a hierarchal structure that identifies the relationships between the meta-model elements; and
(2) and the design of the set of criteria to identify the level of granularity of the functional requirements specifications.
The meta-model and the set of criteria are designed in two variants:

- the 1st variant captures the functional components of a software application without specifying how these functional components are documented; and
- the 2nd variant captures the functional components of a software application by using elements from the COSMIC measurement method and the UML use-case model.

In the subsequent chapters, Chapter 6 will present the evaluation of the usability of the scaling factors framework with three (3) different groups of participants with different experience profiles. Chapter 8 will present the evaluation of the applicability of the scaling factors framework with four (4) case studies that document the functional requirements specifications of software applications that are different in the software type, business objective, and context.

CHAPTER 6

VERIFICATIONS OF THE USABILITY OF THE SCALING FACTORS FRAMEWORK

6.1 Introduction

This chapter includes conducting three (3) experimental sessions to apply the scaling factors framework with the same case study by three (3) different groups of industry practitioners. In this chapter, the participants were asked to assign scaling factors to the same set of non-complete, non-detailed software requirements specifications by applying the scaling factors framework, in order to identify and rank the level of granularity of these functional requirements specifications.

This chapter is organized as follows: Section 6.2 presents the context of the SRS document used in the experimentation. Section 6.3 presents the design of the experiment steps. Section 6.4 presents the experimental results of the experiment. Section 6.5 presents potential threats to validity. And finally, a discussion of the experiment and suggestions for future directions are presented in the chapter summary in Section 6.6.

6.2 The Software Requirements Specifications Document

The software requirements specifications (SRS) document (GÉLOG, 2008) used in the three (3) experimental sessions is written in accordance to the Rational Unified Process (RUP) (Kruchten, 2000) in terms of contents and structure, as an example of an online course registration system that was developed for Wylie College in (RUP, 1999).

The SRS document (GÉLOG, 2008) consists of (14) pages of textual specification, divided into two (2) main sections:

- section 1 provides introductory information, including background information, software purpose and scope, software objectives, and references; and

- section 2 provides description of the software functionality and features including functional requirements, the characteristics of the users, constraints, and assumptions.

The functional size of this SRS document was measured using the international standard for software functional size measurement: COSMIC (ISO19761, 2011). Its functional size of 107 COSMIC Function Point was measured and documented as a COSMIC case study by a team of eight (8) measurement experts: with an average of fifteen (15) years of industrial experience, were COSMIC Certified Entry Level practitioners (GÉLOG, 2013), were experienced in functional size measurement, and were active members of the COSMIC Measurement Practice Committee at that time.

The team of eight (8) measurement experts who measured the SRS document (GÉLOG, 2008) has made measurements assumptions during the measurement process, in order to clarify all the ambiguities that may exist in the SRS document. These measurement assumptions/clarifications were added by the principal researcher into their associated functional specifications in the SRS document to reduce the possibility of misunderstanding of the specifications by the participants in the experiment.

The SRS document (GÉLOG, 2008) describes the functionality of the course registration system using nine (9) use-cases: these use-cases include thirty-six (36) use-case scenarios to specify the registration system functionality in textual form. The original SRS document (GÉLOG, 2008) was modified by the principal researcher in order to adapt it to the objective of this experiment (i.e. assign scaling factors to a set of non-complete, non-detailed software requirements specifications using the scaling factors framework).

The reason for the document modification is that the original SRS document (GÉLOG, 2008) was considered to be detailed and complete – especially after adding the measurement assumptions by the team of experts – and it specifies in detail the functionality that has to be delivered by the course registration system: it allows a standardized functional size

measurement method to be used to obtain an accurate measurement of the software system functional size.

The principal researcher has applied the proposed scaling factors framework into the original SRS document and obtained the following results:

- nine (9) use-cases were assigned a *'<specified> use-case'* scaling category; and
- their associated thirty-six (36) use-case scenarios were assigned a *'<fully-specified> scenario'* scaling category to indicate their levels of granularity.

For the purpose of experimentation, the original SRS document was modified as follows:

- one (1) use-case was kept 'as is' (i.e. without any modification of it specification);
- three (3) use-cases were partially modified, by removing some portions of their specifications; and
- five (5) use-cases were completely modified by removing use-cases specifications entirely.

The principal researcher applied next the scaling factors framework into the modified version of the SRS document: table 6.1 presents the reference assignment of the scaling factors to the nine (9) use-cases and their associated use-case scenarios: the results after the modifications are presented at the use-case level:

- four (4) use-cases were assigned a *<specified> use-case* scaling category;
- three (3) use-cases were assigned a *<unspecified> use-case* scaling category; and
- two (2) use-cases were assigned a *<implicit> use-case* scaling category.

On the other hand, at the scenario level:

- four (4) scenarios were assigned a *<fully-specified> scenario* scaling category;
- two (2) scenarios were assigned a *<partially-specified> scenario* scaling category;
- eight (8) scenarios were assigned a *<unspecified> scenario* scaling category; and
- seven (7) scenarios were assigned a *<implicit> scenario* scaling category; and

Table 6.1 The scaling factors framework applied to the modified version of the SRS

Use Case Title	**Scaling Factor**
1. Login	<Specified> use-case
Main flow: validation of name/password	<Fully-specified> scenario
Alternative flow: invalid name/password	<Fully-specified> scenario
2. Maintain professor information	<Specified> use-case
Main flow: Add professor	<Fully-specified> scenario
Alternative flows: Professor already exist	<Unspecified> scenario
Modify professor	<Implicit> scenario
Delete professor	<Implicit> scenario
Professor Not found	<Unspecified> scenario
3. Select/de-select courses to teach	<Unspecified> use-case
4. Maintain student information	<Specified> use-case
Main flow: Add student	<Partially-specified> scenario
Alternative flows: Student already exist	<Unspecified> scenario
Modify student	<Implicit> scenario
Delete student	<Implicit> scenario
Student not found	<Unspecified> scenario
5. Register for courses	<Specified> use-case
Main Flow: Create schedule	<Fully-specified> scenario
Alternative flows: Course registration closed	<Implicit> scenario
Catalogue sys. unavailable	<Implicit> scenario
Add course offering	<Partially-specified> scenario
Un-fulfilled pre-requisite Or course full	<Unspecified> scenario
Save schedule	<Unspecified> scenario
Modify schedule	<Unspecified> scenario
Delete schedule	<Unspecified> scenario
No schedule found	<Implicit> scenario
6. Monitor course full	<Unspecified> use-case
7. Close registration	<Unspecified> use-case
8. Submit grades	<Implicit> use-case
9. View report card	<Implicit> use-case

After that, the principal researcher asked an independent expert (i.e. Expert #B6) with an average of fifteen (15) years of experience in the software engineering industry and ten (10) years of industrial experience in applying the international standard for software functional

size measurement: COSMIC (ISO19761, 2011) to apply the scaling factors framework into the original and modified versions of the SRS document in order to verify the correctness of the assignment made by the principal researcher. The independent expert (i.e. Expert #B6) has confirmed the correctness of the assignments of the scaling categories made by the principal researcher.

6.3 Steps of Experimentation

The activities in the experimental design were performed in their chronological order, where the outputs of one activity were used as inputs to subsequent activities, as required:

6.3.1 Experiment preparation

This activity consisted of two sub activities: experiment kit preparation, and experiment pilot testing, as follows:

1) **Experiment Kit Preparation:** prior to the three (3) experimental sessions, the proposed scaling factors framework was reviewed by the principal researcher in order to prepare for the participants in the experiment a clear description of the scaling factors framework, and a set of rules and instructions to follow during the experiment session. The experiment kit (See Appendix III on the CD attached to this thesis) was provided to the participants in the three (3) experimental sessions in a pre-experiment training session. The experiment kit included:
   - **A description of the scaling factors framework:** the participants in the experiment were provided with a full-description of the proposed scaling factors framework; this includes description of both variants of the set of criteria and their associated meta-models.

   - **The software requirements specifications document:** the participants in the experiment were provided with the modified version of the SRS document, which

includes a reserved space beside each use-case/use-case scenario for scaling factors assignment.

- **A defined set of rules:** the participants in the experiment were provided with a set of rules that governs the participants' activities during the experiment session; this includes communication restrictions with other participants in the experiment, signing the participation consent form, and the physical storage location of the experimental data.

- **The role of participants:** the participants' role in the experiment was fully-explained in the training session. For example, they are taking the role of what the requirements engineer is supposed to do: assigning scaling factors using a well-defined methodology to software requirements specifications with different levels of granularity at early phases of the software development life cycle.

2) **Experiment Pilot Testing:** a preliminary run of the experiment was conducted by the principal researcher and an independent expert in the software engineering industry: this preliminary run of the experiment is to identity early the potential challenges in the activities of the experiment, including:
   a. the applicability of the SRS document to the experiment, in terms of objective and scope;
   b. an estimate of the time required to conduct the experiment;
   c. the usability of the data collection forms to be used in the participants in the experiment; and
   d. verify the correctness of the reference assignment (see table 6.1) of the scaling factors using the proposed scaling factors framework conducted by the independent expert (i.e. Expert #B6) in the software engineering field.

6.3.2 Call for participation

The experiment was conducted in three (3) separate sessions; two sessions included graduate students, mostly from industry, enrolled in two different graduate courses in software engineering (MGL800: Project management & MGL841: Measurement: a key concept in software engineering) offered at the École de technologie supérieure in the summer of 2012, and the 3rd experiment session included attendees at the Joint Conference of the 22nd International Workshop on Software Measurement (IWSM) and the 7th International Conference on Software Process and Product Measurement (Mensura) in October, 17 2012. All the participants who volunteered to participate in the experiment had to indicate their experience in various software engineering disciplines.

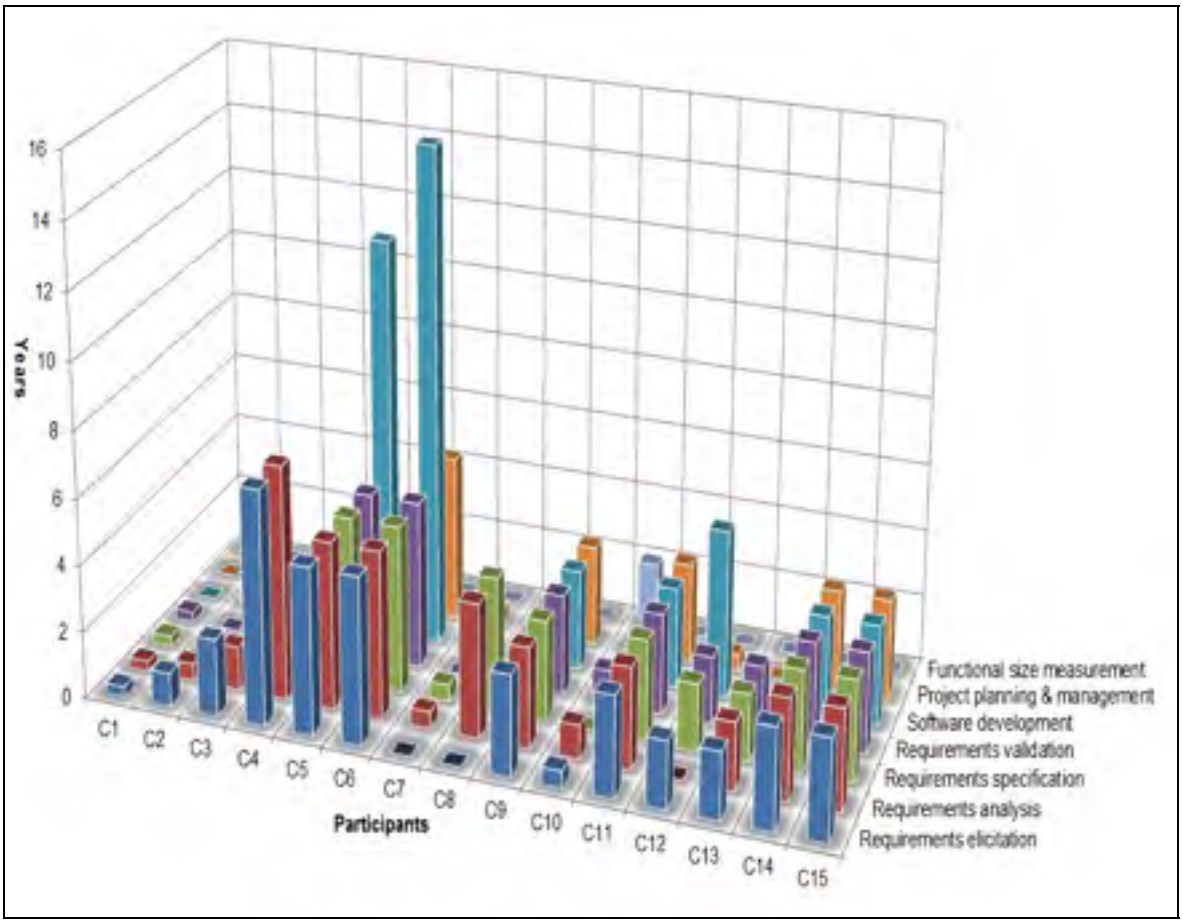

Figure 6.1 Group #1 – participants in a project management graduate-level course

The industrial experience profiles of the three (3) groups of participants involved in the three (3) experimental sessions are presented in figure 6.1, figure 6.2, and figure 6.3:

- figure 6.1 presents Group #1 which includes the 15 participants in the project management graduate-level course;
- figure 6.2 presents Group #2 which includes the 10 participants in software measurement graduate-level course; and
- figure 6.3 presents Group #3 which includes the 16 participants who attended the joint conference of IWSM-Mensura 2012.

These experience profiles are structured in accordance to the participants' levels of experience in various software engineering topics. It is worth mentioning that the participants in Group #3 have significant experience in software functional size measurement, and software effort and cost estimation, whereas, the participants Group #1 & Group #2 do not have such measurement experience.

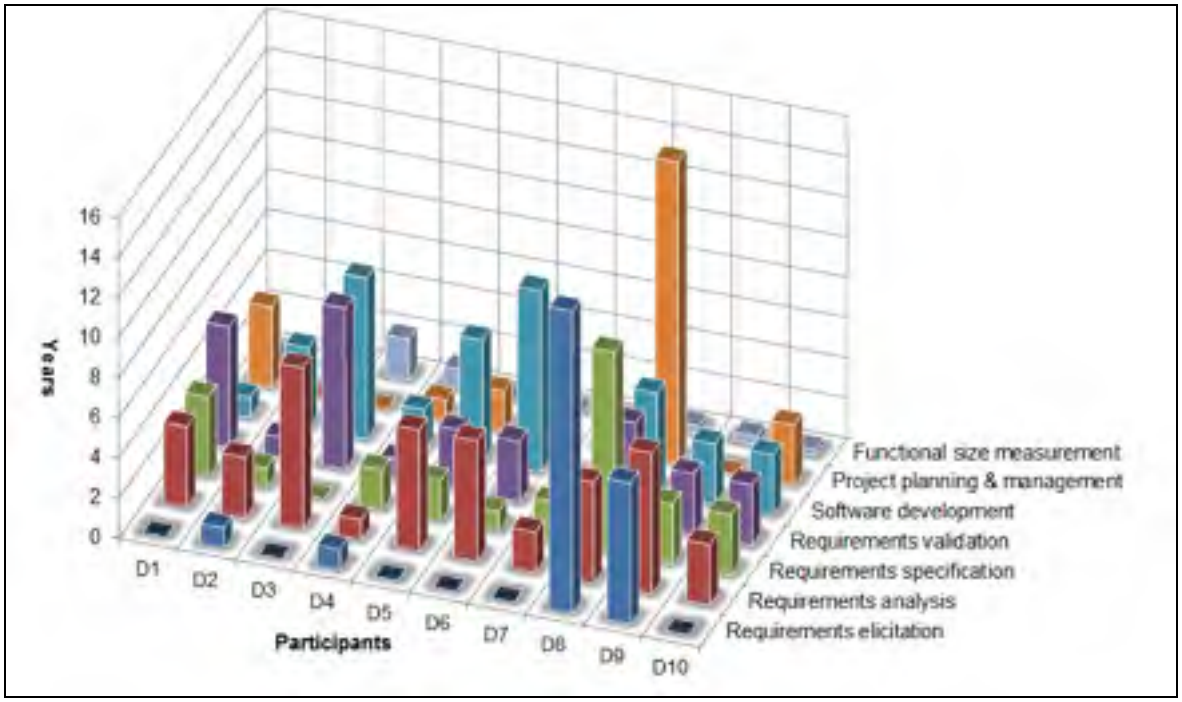

Figure 6.2 Group #2 – participants in a software measurement graduate-level course

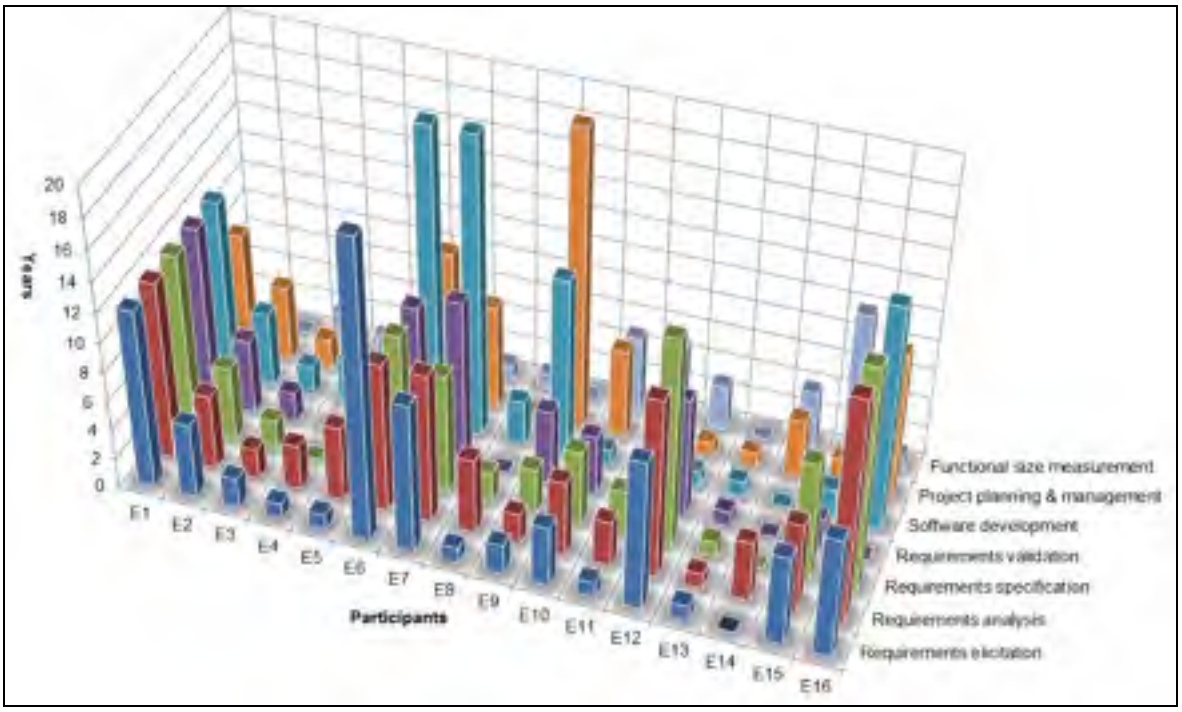

Figure 6.3 Group #3 – participants in IWSM-Mensura 2012

6.3.3 Pre-experiment training

The participants in the three (3) experimental sessions were given a one-hour training session, to familiarize them with the concepts presented in the scaling factors framework, the SRS document, the rules to follow, and their role that governs their behavior while conducting the activities of the experiment. The participants were also given fifteen (15) minutes to read the SRS document.

6.3.4 Conducting the experiment

The participants in the three (3) experimental sessions were given sixty (60) minutes to assign scaling factors using the proposed scaling factors framework. More specifically, each participant in the three (3) experimental sessions was handled a printed copy of the SRS document, and asked to fill in the following data in a space reserved next to each use-case/use-case scenario:

- the scaling factor for each use-case/use-case scenario; and
- total effort spent to assign the scaling factors using the scaling factors framework.

6.4 The Experimental Results

The term "experimental results" in this context refers to the data collected during the three (3) experimental sessions after being analyzed for consistency, validity, and free of outlier data. The following data was collected during the experimental sessions:

- the experience of the participants in various software engineering disciplines, including requirements analysis, requirements specifications, requirements validation, and software functional size measurement;
- the assignment of the scaling factors to the requirements specifications explained in the software requirements specifications (SRS) document, using the scaling factors framework; and
- the total effort required to conduct each activity in the experiment.

6.4.1 Identification of the levels of granularity

Table 6.2 and table 6.3 present the assignment of the scaling factors to the software requirements specifications (i.e. use-cases and use-case scenarios) using the scaling factors framework by the fifteen (15) participants in Group #1:

- table 6.2 presents the assignment of participants (C1) – (C8); and
- table 6.3 presents the assignment of participants (C9) – (C15).

Both tables present the list of use-case/use-case scenarios that describe the functionality of the course registration system presented in the SRS document. Also, they present the correct assignment of the scaling factors that was prepared by the principal researcher and verified by the independent expert (i.e. Expert #B6). It is worth mentioning that a tick (√) mark indicates that a participant has identified the correct level of granularity and (X) mark

indicates that the participant has identified the wrong level of granularity at the use-case level or at the scenario level.

Table 6.2 Scaling factors assignment by participants (C1) – (C8) in Group #1

#	Use-case title	Correct scaling category	Participants Assignment							
			C1	C2	C3	C4	C5	C6	C7	C8
1	Login	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓	✓
1.1	Validation of name/password	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓
1.2	Invalid name/password	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓
2	Maintain professor information	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓	✓
2.1	Add professor	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓
2.2	Professor already exist	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓
2.3	Professor not found	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓
2.4	Update professor information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓
2.5	Delete professor information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓
3	Select/de-select Courses to teach	<Unspecified use-case>	✓	✓	✓	✓	✓	✓	✓	✗
4	Maintain student information	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓	✓
4.1	Add student	<Partially-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓
4.2	Student already exist	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓
4.3	Student not found	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓
4.4	Update student information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓
4.5	Delete student information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓
5	Register for courses	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓	✓
5.1	Create schedule	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓
5.2	Save schedule	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓
5.3	Modify schedule	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓
5.4	Delete schedule	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓
5.5	Register course offering	<Partially-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓
5.6	Unfulfilled prerequisite or course full	<Unspecified scenario>	✓	✓	✓	✗	✓	✗	✓	✓
5.7	Check if registration closed	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓
5.8	Check if course catalog available	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓
5.9	Retrieve student schedule information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓
6	Close registration	<Unspecified use-case>	✓	✓	✓	✓	✓	✓	✓	✗
7	Submit grades	<Unspecified use-case>	✓	✗	✓	✓	✓	✓	✓	✗
8	View course information	<Implicit use-case>	✓	✓	✓	✓	✓	✓	✓	✓
9	Check course status	<Implicit use-case>	✓	✓	✓	✓	✓	✓	✓	✓

The participants in Group #1 have an average of five (5) years of experience in the software engineering industry including, software requirements analysis and specification, requirements validation, and software projects planning and management (see figure 6.1 for participants experience profile). But, they have no academic and industrial experience in software functional size measurement.

The assignment of the scaling factors to requirements specifications described in the use-case/use-case scenarios in the SRS document were analyzed and then compared with the reference assignment (see table 6.1) prepared by the principal researcher.

All the (15) participants in Group #1 were able to correctly identify the correct number of all the use-cases and their associated use-case scenarios when they applied the scaling factors framework to the SRS document;

Table 6.3 Scaling factors assignment by participants (C9) – (C15) in Group #1

#	Use-case title	Correct scaling category	Participants Assignment						
			C9	C10	C11	C12	C13	C14	C15
1	**Login**	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓
1.1	Validation of name/password	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓
1.2	Invalid name/password	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓
2	**Maintain professor information**	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓
2.1	Add professor	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓
2.2	Professor already exist	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓
2.3	Professor not found	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓
2.4	Update professor infromation	<implicit scenario>	✓	✓	✓	✓	✓	✓	✓
2.5	Delete professor information	<implicit scenario>	✓	✓	✓	✓	✓	✓	✓
3	**Select/de-select Courses to teach**	<Unspecified use-case>	✓	✓	✓	✓	✓	✓	✓
4	**Maintain student information**	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓
4.1	Add student	<Partially-specified scenario>	✓	✓	✓	✓	✓	✓	✓
4.2	Student already exist	<Unspecified scenario>	✗	✓	✓	✓	✓	✓	✓
4.3	Student not found	<Unspecified scenario>	✗	✓	✓	✓	✓	✓	✓
4.4	Update student information	<implicit scenario>	✓	✓	✓	✓	✓	✓	✓
4.5	Delete student information	<implicit scenario>	✓	✓	✓	✓	✓	✓	✓
5	**Register for courses**	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓
5.1	Create schedule	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓
5.2	Save schedule	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓
5.3	Modify schedule	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓
5.4	Delete schedule	<Unspecified scenario>	✗	✓	✓	✓	✓	✓	✓
5.5	Register course offering	<Partially-specified scenario>	✓	✓	✓	✓	✓	✓	✓
5.6	Unfulfilled prerequisite or course full	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓
5.7	Check if registration closed	<implicit scenario>	✓	✓	✓	✓	✓	✓	✓
5.8	Check if course catalog available	<implicit scenario>	✓	✓	✓	✓	✓	✓	✓
5.9	Retrieve student schedule information	<implicit scenario>	✓	✓	✓	✓	✓	✓	✓
6	**Close registration**	<Unspecified use-case>	✓	✓	✗	✓	✓	✓	✓
7	**Submit grades**	<Unspecified use-case>	✓	✓	✗	✓	✓	✓	✓
8	**View course information**	<implicit use-case>	✓	✓	✓	✓	✓	✓	✓
9	**Check course status**	<implicit use-case>	✓	✓	✓	✓	✓	✓	✓

Eight (8) participants in Group #1 were able to correctly identify the correct levels of granularity for all the use-cases and their associated use-case scenarios when they applied the scaling factors framework to the SRS document.

On the other hand, three (3) participants in Group #1 have committed one (1) miss-assignment of scaling factors when they applied the scaling factors framework to the SRS document. One (1) participant in Group #1 has committed two (2) miss-assignments of scaling factors when he applied the scaling factors framework to the SRS document. Two (2) participants in Group #1 have committed three (3) miss-assignments of scaling factors when they applied the scaling factors framework to the SRS document. One (1) participant in Group #1 has committed six (6) miss-assignments of scaling factors for six (6) use-case scenarios when he applied the scaling factors framework to the SRS document.

Table 6.4 presents a summary of the miss-assignments of the scaling factors to the use-case/use-case scenarios described in the SRS document by the 15 participants in Group #1.

Table 6.4 Summary of scaling factors miss-assignment by participants in Group #1

	Total number of scaling factors miss-assignment				
Participant Code	**0**	**1**	**2**	**3**	**6**
	C1	C2	C15	C8	C9
	C3	C4		C11	
	C5	C6			
	C7				
	C10				
	C12				
	C13				
	C14				

It is worth mentioning that participants in Group #1 who made miss-assignments (i.e. participants who committed a total of 1, 2, 3 or 6 miss-assignments as presented in table 6.4) for some of the use-cases and the use-case scenarios have correctly assigned scaling factors to similar (i.e. at similar levels of granularity) use-cases/use-case scenarios in the SRS document.

For example, participant (C2) in Group #1 has miss-assigned to the ‘submit grades’ use-case an *‘Implicit use-case’* scaling factor while he correctly assigned other use-cases such as the

'close registration' and 'select/de-select courses to teach' an *'unspecified use-case'* scaling factor to rank their level of granularity, even though the three use-cases (i.e. the miss-assigned and the correctly assigned) are at the same level of granularity.

Similarly, participant (C4) in Group #1, has miss-assigned to the 'unfulfilled pre-requisite or 'course full' use-case scenario an *'Implicit use-case scenario'* scaling factor, but correctly assigned the correct scaling factor to all the remaining use-case scenarios in the SRS document. Also, participant (C6) committed the same miss-assignment committed by participant (C4).

Further, participant (C9) in Group #1, who miss-assigned six (6) use-case scenarios as an *'Implicit use-case scenarios'*, whereas, he assigned the correct scaling factor to similar (i.e. at similar levels of granularity) use-case scenarios like 'save schedule' and 'modify schedule' use-case scenarios.

Table 6.5 Scaling factors assignment by the 10-participants in Group #2

#	Use-case title	Correct scaling category	Participants Assignment									
			D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
1	**Login**	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.1	Validation of name/password	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.2	Invalid name/password	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2	**Maintain professor information**	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.1	Add professor	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.2	Professor already exist	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.3	Professor not found	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.4	Update professor information	<Implicit scenario>	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓
2.5	Delete professor information	<Implicit scenario>	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓
3	**Select/de-select Courses to teach**	<Unspecified use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
4	**Maintain student information**	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
4.1	Add student	<Partially-specified scenario>	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓
4.2	Student already exist	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
4.3	Student not found	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
4.4	Update student information	<Implicit scenario>	✗	✓	✓	✓	✗	✗	✓	✗	✓	✓
4.5	Delete student information	<Implicit scenario>	✗	✓	✓	✓	✗	✗	✓	✗	✓	✓
5	**Register for courses**	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.1	Create schedule	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓
5.2	Save schedule	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.3	Modify schedule	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.4	Delete schedule	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.5	Register course offering	<Partially-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.6	Unfulfilled prerequisite/course full	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.7	Check if registration closed	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.8	Check if course catalog available	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.9	Retrieve student schedule info.	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
6	**Close registration**	<Unspecified use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
7	**Submit grades**	<Unspecified use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
8	**View course information**	<Implicit use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
9	**Check course status**	<Implicit use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 6.5 presents the assignment of the scaling factors to the software requirements specifications using the proposed scaling factors framework by the ten (10) participants in Group #2. It is worth mentioning that participants in Group #2 have an average of eight (8) years of experience in the software engineering industry including, software requirements analysis and specification, requirements validation, and software projects planning and management (see figure 6.2 for their experience profile).

The assignment (by the participants in Group #2) of the scaling factors to requirements specifications described in the use-case/use-case scenarios in the SRS document were analyzed and then compared with the reference assignment (see Table 6.1) prepared by the principal researcher:

- all the ten (10) participants in Group #2 were able to correctly identify all the use-cases and their associated use-case scenarios when they applied the scaling factors framework to the SRS document;
- five (5) participants in Group #2 were able to correctly identify the level of granularity of all the use-cases and their associated use-case scenarios when they applied the scaling factors framework to the SRS document;
- one (1) participant in Group #2 have committed two (2) miss-assignments of scaling factors for two (2) use-case scenarios when he applied the scaling factors framework to the SRS document; and
- four (4) participants in have committed four (4) miss-assignments of scaling factors for six (6) different use-case scenarios when they applied the scaling factors framework to the SRS document.

Table 6.6 presents a summary of the miss-assignments of the scaling factors to the use-case/use-case scenarios described in the SRS document by participants in Group #2. It is worth mentioning that participants in Group #2 who made miss-assignments (i.e. participants who made a total of 1, 2, 3 or 6 miss-assignments as presented in table 6.6) for some of the use-case scenarios have correctly assigned scaling factors to similar (i.e. at similar levels of granularity) use-case scenarios in the SRS document.

Table 6.6 Summary of scaling factors miss-assignment by participants in Group #2

Participant Code	Total number of scaling factors miss-assignment		
	0	**2**	**4**
	D2	D1	D5
	D3		D6
	D4		D7
	D9		D8
	D10		

Table 6.7 Total effort expended by the 25-participants in Group #1 and Group #2

Group Code		Participant Code	Effort (in minutes)	Average effort
	#1	C1	35	41 minutes
		C2	41	
		C3	50	
		C4	22	
		C5	60	
		C6	55	
		C7	15	
		C8	20	
		C9	60	
		C10	38	
		C11	45	
		C12	60	
		C13	55	
		C14	30	
		C15	27	
	#2	D1	60	49 minutes
		D2	40	
		D3	30	
		D4	60	
		D5	55	
		D6	45	
		D7	60	
		D8	33	
		D9	55	
		D10	47	

For example, participant (D1) in Group #2 has miss-assigned 'update/delete student information' use-case scenarios an *'Unspecified use-case scenario'* scaling factor, on the other hand, he correctly assigned similar (i.e. at similar levels of granularity) use-case scenarios like 'update/delete professor information' and 'check if registration closed', 'check if course catalogue available', *'Implicit use-case scenario'* scaling factor to indicate their level of granularity, even though the use-cases scenarios (i.e. the miss-assigned and the correctly assigned) are at the same level of granularity. In similar fashion, the miss-assignments committed by participants (D5), (D6), (D7), and (D8) in Group #2 can be explained.

Table 6.8 Scaling factors assignment by participants (E1) – (E7) in Group #3

#	Use-case title	Correct scaling category	Participants Assignment						
			E1	E2	E3	E4	E5	E6	E7
1	**Login**	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓
1.1	Validation of name/password	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✗
1.2	Invalid name/password	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✗
2	**Maintain professor information**	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓
2.1	Add professor	<Fully-specified scenario>	✓	✗	✓	✓	✓	✓	✓
2.2	Professor already exist	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓
2.3	Professor not found	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓
2.4	Update professor information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓
2.5	Delete professor information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓
3	**Select/de-select courses to teach**	<Unspecified use-case>	✓	✓	✓	✓	✓	✓	✓
4	**Maintain student information**	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓
4.1	Add student	<Partially-specified scenario>	✓	✓	✓	✓	✓	✓	✓
4.2	Student already exist	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓
4.3	Student not found	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓
4.4	Update student information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓
4.5	Delete student information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓
5	**Register for courses**	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓
5.1	Create schedule	<Fully-specified scenario>	✓	✗	✓	✓	✓	✗	✓
5.2	Save schedule	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓
5.3	Modify schedule	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓
5.4	Delete schedule	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓
5.5	Register course offering	<Partially-specified scenario>	✓	✓	✓	✓	✗	✓	✓
5.6	Unfulfilled prerequisite or course full	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓
5.7	Check if registration closed	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓
5.8	Check if course catalog available	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓
5.9	Retrieve student schedule information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓
6	**Close registration**	<Unspecified use-case>	✓	✓	✓	✓	✓	✓	✓
7	**Submit grades**	<Unspecified use-case>	✓	✓	✓	✓	✓	✓	✓
8	**View course information**	<Implicit use-case>	✓	✓	✓	✓	✓	✓	✓
9	**Check course status**	<Implicit use-case>	✓	✓	✓	✓	✓	✓	✓

The participants in the three (3) experimental sessions were given forty-five (45) minutes to apply the scaling factors framework to the software requirements specifications explained in the SRS document. Table 6.7 presents the total effort expended by the participants in Group #1 and Group #2: each participant had to indicate the start and end times for applying the proposed framework.

Table 6.8 and table 6.9 present the assignment of the scaling factors to the software requirements specifications using the proposed scaling factors framework by the participants in Group #3 (i.e. the attendees of IWSM-Mensura 2012). Table 6.8 presents the assignment of the scaling factors by participants E1 – E7, while table 6.9 presents the assignment of the scaling factors by participants E8 – E1. It is worth mentioning that the assignment of the scaling factors by participants in Group #3 was separated in two tables for ease of presentation).

Table 6.9 Scaling factors assignment by participants (E8) – (E16) in Group #3

#	Use-case title	Correct scaling category	Participants Assignment								
			E8	E9	E10	E11	E12	E13	E14	E15	E16
1	Login	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.1	Validation of name/password	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.2	Invalid name/password	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
2	Maintain professor information	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.1	Add professor	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.2	Professor already exist	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.3	Professor not found	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.4	Update professor information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.5	Delete professor information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
3	Select/de-select Courses to teach	<Unspecified use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓
4	Maintain student information	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓
4.1	Add student	<Partially-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
4.2	Student already exist	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
4.3	Student not found	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
4.4	Update student information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
4.5	Delete student information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
5	Register for courses	<Specified use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.1	Create schedule	<Fully-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.2	Save schedule	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.3	Modify schedule	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.4	Delete schedule	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.5	Register course offering	<Partially-specified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.6	Unfulfilled prerequisite or course full	<Unspecified scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.7	Check if registration closed	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.8	Check if course catalog available	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
5.9	Retrieve student schedule information	<Implicit scenario>	✓	✓	✓	✓	✓	✓	✓	✓	✓
6	Close registration	<Unspecified use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓
7	Submit grades	<Unspecified use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓
8	View course information	<Implicit use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓
9	Check course dates	<Implicit use-case>	✓	✓	✓	✓	✓	✓	✓	✓	✓

The assignment – by the participants in Group #3 – of the scaling factors to requirements specifications described in the use-case/use-case scenarios in the SRS document were analyzed and then compared with the reference assignment (see table 6.1) prepared by the principal researcher:

- all the sixteen (16) participants in Group #3 were able to correctly identify all the use-cases and their associated use-case scenarios when they applied the scaling factors framework to the SRS document;
- twelve (12) participants in Group #3 were able to correctly identify the level of granularity of all the use-cases and their associated use-case scenarios when they applied the scaling factors framework to the SRS document;
- three (3) participants (i.e. E2, E5 and E6) in Group #3 have committed one (1) miss-assignment of scaling factors when they applied the scaling factors framework to the SRS document; and
- one (1) participant (i.e. E7) in Group #3 has committed two (2) miss-assignments of scaling factors when he applied the scaling factors framework to the SRS document.

Table 6.10 Summary of scaling factors miss-assignment by participants in Group #3

Participant Code	**Total number of scaling factors miss-assignment**		
	0	**1**	**2**
	E1	E2	E7
	E3	E5	
	E4	E6	
	E8		
	E9		
	E10		
	E11		
	E12		
	E13		
	E14		
	E15		
	E16		

Table 6.10 presents a summary of the miss-assignments of the scaling factors to the use-case/use-case scenarios described in the SRS document by participants in Group #3. It is worth mentioning that participants in Group #3 who committed miss-assignments (i.e. participants E2, E5, E6 and E7) for some of the use-case scenarios have correctly assigned scaling factors to similar (i.e. at similar levels of granularity) use-case scenarios in the SRS document.

For example, participant E2 has miss-assigned 'add professor' and 'create schedule' scenarios a *'partially-specified scenario'* scaling factor. On the other hand, he correctly assigned similar (i.e. at similar levels of granularity) use-case scenarios like 'validation of name/password' and 'invalid name/password' use-case scenarios a *'fully-specified scenario'* scaling factor to indicate their level of granularity, even though the use-cases scenarios (i.e. the miss-assigned and the correctly assigned) are at the same level of granularity.

Similarly, participant E5 and participant E6 have committed the same miss-assignment of scaling factors: participant E5 has miss-assigned 'register course offering' scenario an 'unspecified scenario' scaling factor, even though this scenario contains few details that represent 'partial specification' for such scenario. On the other hand, participant E5 has correctly assigned similar (i.e. at similar levels of granularity) scenarios like 'add student' a '*partially-specified scenario'* scaling factor' to indicate their level of granularity, even though the use-cases scenarios (i.e. the miss-assigned and the correctly assigned) are at the same level of granularity.

Furthermore, participant E7 has miss-assigned the two (2) scenarios that are associated with the 'Login' use-case a 'partially-specified scenario', whereas, he correctly assigned other scenarios like 'add professor' the correct scaling factor to rank its level of granularity.

Table 6.11 presents the total effort expended by the participants in Group #3: each participant had to indicate the start and end times for applying the proposed framework. The effort data presented in table 6.11 shows that all the participants in Group #3 have completed the

workshop activities within the pre-defined time by the principal researcher (As presented in table 6.11 the average effort is forty (40) minutes for the sixteen participants in Group #3).

Table 6.11 Total effort expended by the 16-participants in Group #3

		Participant Code	Effort (in minutes)	Average effort
Group Code	#3	E1	48	40 minutes
		E2	25	
		E3	50	
		E4	35	
		E5	42	
		E6	45	
		E7	50	
		E8	35	
		E9	43	
		E10	40	
		E11	50	
		E12	25	
		E13	25	
		E14	37	
		E15	50	
		E16	45	

6.4.2 Summary of findings

All the participants in the three (3) experimental sessions were able to identify the correct number of all the use-cases and their associated use-case scenarios. In terms of identifying the levels of granularity:

- sixty-one (61) percent of participants made zero (0) mistakes in identifying the levels of granularity;
- fifteen (15) percent of participants made one (1) mistake in identifying the levels of granularity;
- seven (7) percent of participants made two (2) mistakes in identifying the levels of granularity;

- five (5) percent of participants made three (3) mistakes in identifying the levels of granularity;
- ten (10) percent of participants made four (4) mistakes in identifying the levels of granularity; and
- two (2) percent of participants made six (6) mistakes in identifying the levels of granularity.

These miss-assignments made by the three (3) groups of participants may be due to the time to conduct the experiment which was limited to the duration of the experiment session. Therefore, the participants did not have the opportunity to ask for clarifications from their colleagues, or experts in the field which is common practice in the software development organizations.

On the other hand, in a previous experiment (See Chapter 4), only one (1) participant out of twelve (12) participants was able to correctly identify the correct number of use-cases. However, this participant was not able to identify and then rank the correct levels of granularity of such identified use-cases using only the rules and concepts of the *Early & Quick* COSMIC technique. This is due to the inexistence of guidelines to take into consideration the preliminary steps recommended in (COSMIC, 2007).
In summary, the experimental results collected and analyzed in this chapter from the three (3) experimental sessions show that the proposed scaling factors framework is usable by three (3) different groups of practitioners with different experience profiles in the software engineering industry.

6.5 Validity Threats

6.5.1 Construct validity threats

A construct validity threat is associated to the validity of the experiment settings to reflect the subject under study (i.e. the usability of the scaling factors framework). The principal

researcher did not perform a check for the quality (for example, ambiguity, consistency, and completeness) (IEEE, 1998) of the software requirements specifications document (GÉLOG, 2008) that was used in the three (3) experimental sessions.

On the other hand, the team of measurement experts who measured the functional size of this requirements specifications document (GÉLOG, 2008) has conducted a preliminary inspection for the ambiguity and the consistency properties while performing the measurement process of the requirements document: the team of measurement experts has provided in (GÉLOG, 2008) clarifications and assumptions to interpret the functionality for some of the requirements specifications in the document. These clarifications and assumptions where taken into consideration by the principal researcher while preparing the requirements specifications document modified (GÉLOG, 2008) for this experimentation.

6.5.2 Internal validity threats

An internal validity threat is associated to the validity of the experimental results to changes in the design of the experiment, such as lack of discussion or clarifications during the three (3) experimental sessions, the lack of clear data collection procedures, or description of the concept(s) that the principal researcher wish to evaluate.

To mitigate the risk of these potential threats, a one (1) hour tutorial session was held prior to the three (3) experimental sessions to describe objectives, scope, rules and the roles of the participants: the principal researcher explained in details the proposed scaling factors framework, and opened the door for discussion to clarify the activities, and the materials of the experiment, including complete description of the scaling factors framework, participant experience survey, and data collection forms.

Moreover, the principal researcher conducted a pilot test (i.e. to conduct the activities of the experiment prior to the three experiment sessions) of the experiment using the help of an independent expert in order to identify any potential challenges in the experiment procedures,

including the applicability of the SRS to the experiment, the time required to conduct the experiment, the usability of the data collection forms to be used by the participants in the experiment, and verify the correctness of the reference assignment of scaling factors to the requirements specifications described in the SRS document. The independent expert has fifteen (15) years of experience in requirements analysis and modeling, and ten (10) years of experience in applying the international standard for software functional size measurement: COSMIC (ISO19761, 2011).

Furthermore, the independent expert (i.e. Expert #B6) is a COSMIC Certified Entry Level practitioner and an active member of the COSMIC Measurement Practice Committee. The independent expert has correctly identified all the use-cases and their associated use-case scenarios. Also, the independent expert (i.e. Expert #B6) was able correctly identify the level of granularity of all the use-cases and their associated use-case scenarios when applying the scaling factors framework to the SRS document.

6.5.3 External validity threats

An external validity threat is associated to the validity to generalize the experimental results obtained during the three (3) experimental sessions outside the experiment settings. The number of the participants in the three (3) experimental sessions was limited to fifteen (15), ten (10), and sixteen (16) participants, respectively, for a total of forty-one (41) participants. However, the three (3) experimental sessions included participants with three (3) profiles: participants with an average of five (5) years, eight (8) years, and eleven (11) years of experience industry experience in information systems development and software engineering.

6.6 Chapter Summary

This chapter has presented the verification of the usability of the scaling factors framework by three (3) groups of practitioners: it included assigning scaling factors to one set of non-

complete and non-detailed software requirements specifications in order to identify their level of granularity at early phases of the software development life cycle using the scaling factors framework.

The scaling factors framework presents a new methodology to take into consideration the preliminary steps recommended in (COSMIC, 2007) for which the authors in (Meli, 1997), (Conte, Iorio et Santillo, 2004) provided no details to the readers on how to apply these steps in practice. The scaling factors framework is designed to transform the identification process of the level of granularity of early requirements specifications into an objective process at the early phases of the software development life cycle.

Another research phase will include additional experiments to verify the applicability (See Chapter 8) of the scaling factors framework using a variety of case studies that represent software applications that are different in software type, business objective, and context. Such experiments are needed to observe the impact of identifying the correct levels of granularity using the scaling factors framework on approximate sizing.

CHAPTER 7

IMPROVING THE DESIGN OF THE SCALING FACTORS FRAMEWORK

7.1 Introduction

This chapter presents the definition of the interval scaling factors which will be assigned by the scaling factors framework to rank the level of granularity of the functional requirement specifications of software development projects using the measurement unit of the international standard for software functional size measurement: COSMIC (ISO19761, 2011). Such definition of the interval scaling factors will help the requirements engineers to calculate an early approximation of the functional size of a software project at the early stages of the software development life cycle.

This chapter is organized as follows: Section 7.2 presents the analysis of the related literature about the interval scaling factors. Section 7.3 presents the definition of the set of interval scaling factors that will be assigned by the scaling factors framework. A chapter summary is presented in Section 7.4.

7.2 Interval scaling factors in the literature

Few studies have been documented in the literature to assign interval scaling factors. The COSMIC guideline *"Advanced & Related Topics version 3.0"* (COSMIC, 2007) presents four (4) approaches aimed to help the personnel involved in the approximation of software functional size at the early stages of the software development life cycle:

1) **Average Functional Process Approach:** this approach approximates the size of new piece of software in two steps:
   A. Sampling and calculation of an average functional process: this step includes:
   - the identification of the functional processes in other pieces of software with similar characteristics to the software to be approximated;

- measuring accurately the functional size of the identified functional processes using the COSMIC measurement method; and
- calculation of the average functional size value of these functional processes. This average functional size value is considered the scaling factor.

B. Approximation of the piece of software: this step includes:

- the identification of the number of functional processes in the piece of software intended to be approximated; and
- multiplication of this number by the average functional process value calculated in Step (A) to compute an approximation of the piece of software.

2) **Fixed Size Classification Approach:** this approach approximates the functional size of new piece of software by:
   - first identifying the functional processes in that piece of software; and
   - classifying them in accordance to their size into one of three or more size band (for example, Small, Medium and Large) in which each band is assigned a corresponding scaling factor.
   - to calculate the functional size approximation:
     - the number of functional processes in each band is arithmetically multiplied by its corresponding band scaling factor; and
     - the functional size of each band is summed to result in an approximation of the piece of software.

3) **Equal Size Bands Approach:** this approach approximates the functional size of software by refining the "Fixed size classification" approach by defining the boundaries for each size band so that the functional size of each band equally contributes to the total software functional size.

4) **Average Use Case Approach:** this approach approximates the functional size of software at a higher level than the functional process level (i.e. the use case level):

- the approach identifies the average number of functional processes in a sample of use cases and then measures their functional size. For example, a use case consists on average (3.5) functional processes and each of average of 6 CFP;
- the functional size of an average use case (i.e. the scaling factor) is equal to 3.5 × 6 CFP = 21 CFP. Therefore, if the new piece of software consists of 14 use cases, the approximated functional size of this software is calculated as 21× 14 = 294 CFP.

A Few studies have proposed interval scaling factors using the "Equal size bands" approach to calculate an approximation of software functional size. For instance:

a) Study of Robobank and Avionics Defence Contractor (Vogelezang et Prins, 2007): this study included the analysis and calculation of an average functional process in four (4) bands / quartiles (Small, Medium, Large and Very Large) of eleven (11) projects from Robobank – a large bank in Netherlands – and one (1) real-time avionics system from an avionics defence contractor.

Table 7.1 presents the average functional process (in CFP) in the four bands in the research study of Robobank and table 7.2 presents the average functional process (in CFP) in the four bands in the research study of the avionics defence contractor.

Table 7.1 Average functional process for each band in 11 projects in Robobank
source: (Vogelezang et Prins, 2007)

Band size	**Average Functional Process Size (in CFP)**
Small	4.0
Medium	6.2
Large	10.8
Very Large	24.7

Table 7.2 Average functional process for each quartile in a real-time avionics system
source: (Vogelezang et Prins, 2007)

Band size	**Average Functional Process Size (in CFP)**
Small	5.5

Table 7.2 Average functional process for each quartile in a real-time avionics system source: (Vogelezang et Prins, 2007) (Continued)

Band size	Average Functional Process Size (in CFP)
Medium	10.8
Large	18.1
Very Large	38.8

b) Study of Business Application Development Projects from (Vogelezang et Prins, 2007): this study included the analysis and calculation of an average functional process in four bands (Small, Medium, Large and Very Large) from thirty-seven (37) business application development projects each of total functional size more than (100 CFP) and they are obtained from twenty-five (25) different companies. These thirty-seven (37) software development projects are measured by an experienced measurer and reviewed by another experienced measurer (Vogelezang et Prins, 2007). Table 7.3 presents the average functional process (in CFP) in each size band.

Table 7.3 Average functional process for each size band in the thirty-seven (37) software development projects - source: (Vogelezang et Prins, 2007)

Band size	Average Functional Process Size (in CFP)
Small	4.8
Medium	7.7
Large	10.7
Very Large	16.4

In addition, several other studies like (Gencel et Bideau, 2012), (Abualkishik et al., 2012), (Juan J. Cuadrado-Gallego, 2010), (Desharnais, Abran et Cuadrado, 2006), (Abran, Desharnais et Aziz, 2005) have presented the measurement using COSMIC method (ISO19761, 2011) and the Function Points Analysis (FPA) method (IFPUG20926, 2009) of different software development projects with the aim to build conversion models that convert the measurement results from the Function Points Analysis method to measurement results in measurement units of the 2nd generation of functional size measurement methods (i.e. COSMIC measurement method (ISO19761, 2011)).

It worth mentioning that only the studies of (Gencel et Bideau, 2012) and (Desharnais, Abran et Cuadrado, 2006) have presented the measurement results at the COSMIC Functional Process level. The study of (Desharnais, Abran et Cuadrado, 2006) included the measurement and analysis of fourteen (14) industrial business applications obtained from a group responsible for the development of and maintenance of software applications from a Canadian Government agency and the study of (Gencel et Bideau, 2012) included the measurement and analysis one portion of a software application that is developed by a large international software development organization in the United Kingdom (UK) for unified reporting. However, the study of (Gencel et Bideau, 2012) has presented the measurement results of only fifteen (15) use cases for the readers.

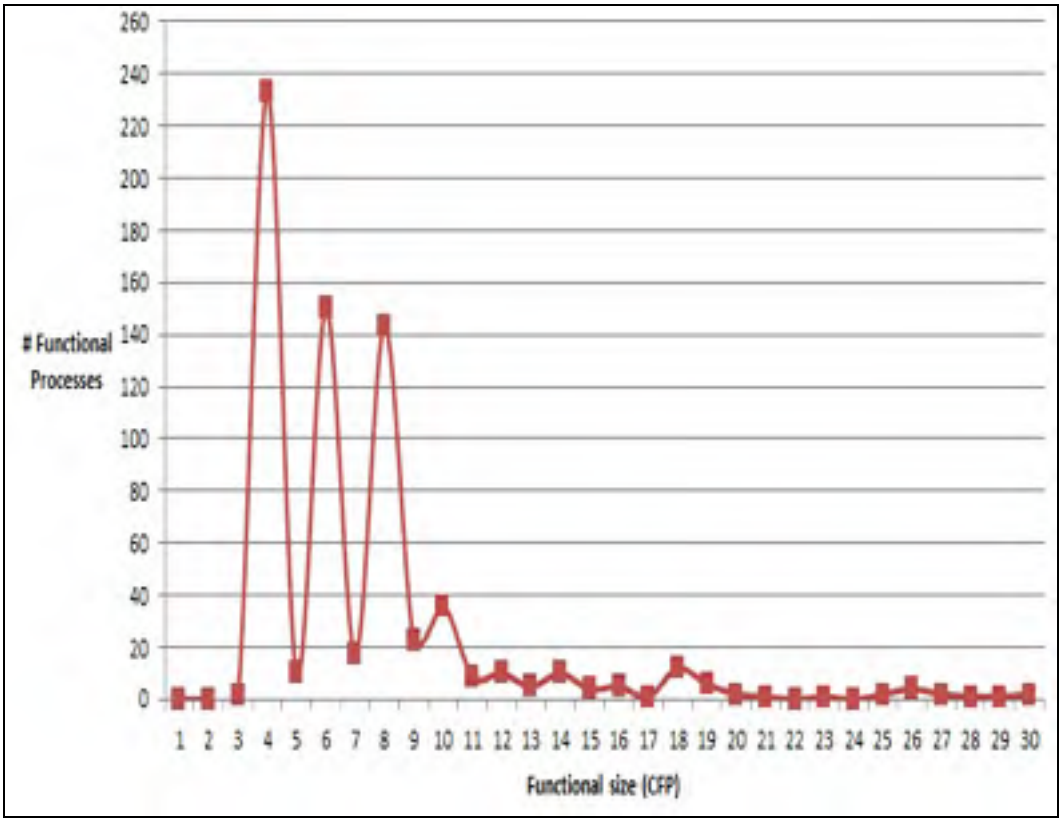

Figure 7.1 Distribution COSMIC functional processes in 15 software applications source: (Gencel et Bideau, 2012), (Desharnais, Abran et Cuadrado, 2006)

Figure 7.1 presents the distribution of the COSMIC functional processes in accordance to their functional size in COSMIC function points (CFP) from both studies of (Gencel et Bideau, 2012) and (Desharnais, Abran et Cuadrado, 2006): the horizontal axes represents the functional size value (in CFP) and the vertical axes represents the number of Functional Processes. For example, the principal researcher has found 233 functional processes in which their measured functional size is 4 CFP in the fifteen (15) software applications.

Table 7.4 Average functional size of functional processes in the 15 software applications source: (Gencel et Bideau, 2012), (Desharnais, Abran et Cuadrado, 2006)

Software Id.	**Number of functional processes**	**Average functional process size (in CFP)**
Study of Desharnais, Abran et Cuadrado (2006)		
#1	16	5.0
#2	20	4.4
#3	19	6.0
#4	11	10.4
#5	50	5.0
#6	53	5.4
#7	49	6.0
#8	45	6.6
#9	53	6.8
#10	53	6.8
#11	52	8.6
#12	67	6.9
#13	96	5.8
#14	82	7.0
Study of (Gencel et Bideau, 2012)		
#15	62	15.1
Minimum		**4.4**
Maximum		**15.1**
Median		**6.6**

Table 7.4 presents the total number of functional processes in each software application and the average *"functional process"* size for each software application. It is worth mentioning

that study of (Vogelezang et Prins, 2007) which involved thirty-seven (37) software applications and the two studies of (Gencel et Bideau, 2012) and (Desharnais, Abran et Cuadrado, 2006) involved the measurement and analysis of software applications of similar type (i.e. business software applications).

7.3 Definition of interval scaling factors in the proposed framework

The objective of this step is to define a set of scaling factors to rank the level of granularity of functional requirements specifications of software projects using the measurement unit of the COSMIC measurement method (ISO19761, 2011).

The set of ordinal scaling factors – at the scenario level – which were presented in both variants of the meta-model and the set of criteria (See Chapter 5) is: (full-specified scenario > partially-specified scenario > unspecified scenario > implicit scenario).

The ordinal scales in this set are organized in a descending order in accordance to the level of granularity they are expected to rank. For example, a 'fully-specified scenario' scale is assigned to a use-case scenario that is explicitly mentioned in the functional user requirements and has a complete functional specification that fully achieves the objective(s) of the software user (See Chapter 5 for the full definition). Whereas, an 'unspecified scenarios' scale is assigned to a use-case scenario that is explicitly mentioned in the functional requirements and contains no details to describe its functional specification.

The initial hypothesis was to propose our own set of interval scaling factors using the analysis of the data (i.e. functional process size values) presented in the two research studies performed by (Gencel et Bideau, 2012) and (Desharnais, Abran et Cuadrado, 2006) through following one of the approaches presented in the *"Advanced & Related Topics version 3.0"* guideline published by the COSMIC group. However, the study of (Desharnais, Abran et Cuadrado, 2006) included only the measurement of fourteen (14) business software applications and the study of (Gencel et Bideau, 2012) has presented the measurement details

of only fifteen (15) use cases (i.e. a portion of the software application). Therefore, this raises concerns with respect to the generic validity if the two studies of (Gencel et Bideau, 2012) and (Desharnais, Abran et Cuadrado, 2006) were used to calculate the interval scaling factors.

After conducting more in-depth research and analysis of previous studies, the principal researcher identified a more promising alternative of using and applying the set of interval scales that were proposed by (Vogelezang et Prins, 2007) (See table 7.3).

The reasons for selecting and using the set of interval scaling factors presented in the study of (Vogelezang et Prins, 2007) are:

- the study includes the measurement and analysis of 37 business software applications, however, the other two studies of (Gencel et Bideau, 2012) and (Desharnais, Abran et Cuadrado, 2006) included the measurement of 14 business software applications and 1 real-time software application;
- the functional size of the 37 software applications is more than 100 CFP; and
- the 37 software applications are obtained from twenty-five (25) different companies over the 2005 – 2006 period.

Each use-case scenario in which its level of granularity is identified as *"fully-specified scenario"* or *"partially-specified scenario"* will be assigned functional size value from table 7.3: (Small = 4.8 CFP, Medium = 7.7 CFP, Large = 10.7 CFP or Very-Large = 16.4 CFP) based on the identification of *"data movements"* in that use-case scenario.

On the other hand, each use-case scenario in which its level of granularity is identified as *"unspecified scenario"* or *"implicit scenario"* will be assigned functional size value from table 7.3: (Small = 4.8 CFP, Medium = 7.7 CFP, Large = 10.7 CFP or Very-Large = 16.4 CFP) based on conducting an analogy-based comparison with similar pieces of software in which the functional size of those pieces of software is accurately measured using the COSMIC measurement method.

It is worth mentioning that only the scenario level is assigned interval scaling factors to rank their level of granularity (i.e. their functional size) using the measurement unit of the COSMIC measurement method. Therefore, the functional size of at the *use-case* level is the aggregation of the functional size of its use-case scenarios and the functional size at the *software component* level is the aggregation of the functional size of its use-cases.

7.4 Chapter Summary

This chapter presented the definition of a set of interval scaling factors to rank the level of granularity of the functional requirements specifications using the measurement unit of the COSMIC measurement method.

This set of interval scaling factors represents the third primary building block of the scaling factors framework, whereas, the first and the second building blocks of the framework are the meta-model and the set of criteria, respectively. The main objective of these two building blocks is to identify the level granularity of the functional requirements specification of software projects that are typically documented at different levels of granularity at the early stages of the software development life cycle.

It is worth mentioning that the usability of the first two building blocks of the scaling factors framework (i.e. the meta-model and set of criteria) has been verified in Chapter 6 with three (3) different groups of practitioners with the same case study (i.e. the C-Registration system). Whereas, the applicability of the scaling factors framework to different case studies has not been verified since it requires a different experimental approach. Further, the third building block of the scaling factors framework (i.e. the interval scaling factors defined in this chapter) has not been verified for usability or applicability. Then, next chapter (i.e. Chapter 8) will present the verification of the applicability of the scaling factors framework with four (4) different case studies: this will also include the verification the applicability of the set of interval scaling factors defined in this chapter.

CHAPTER 8

VERIFICATION OF THE APPLICABILITY OF THE SCALING FACTORS FRAMEWORK

8.1 Introduction

After having presented in the previous chapter the verification of the usability of the proposed scaling factors framework, the next research objective is to verify the applicability of the scaling factors framework. More specifically, this chapter presents the experimentation of the scaling factors framework with four (4) case studies.

In Chapter 6, the objective was to verify the usability of the proposed scaling factors framework by different groups of participants who have had different levels of experience in the software engineering industry and are involved in different phases of the software development life cycle. More specifically, the experimentation for this characteristic of usability of the scaling factors framework has consisted of three (3) different groups of participants in three (3) different experimental sessions using one (1) requirements specifications document (i.e. Course registration system (GÉLOG, 2008).

In this chapter, we present the verification of the applicability of the scaling factors framework to different sets of software requirements specifications: the set of requirements selected represents of four (4) software applications that are different in the software type, business objective, context, and functional size that is measured using the international standard for software functional size measurement – COSMIC (ISO19761, 2011).

This chapter is organized as follows: Section 8.2 presents the experiment design aimed to verify the applicability of the scaling factors framework. Section 8.3 presents the four (4) case studies used for experimentation in this chapter. Section 8.4 presents the experiment preparation. Section 8.5 to 8.8 presents the experimentation with the four (4) case studies. Section 8.9 presents a summary of the chapter findings from the experimentation of the

applicability to the four different sets of documented requirements. Section 8.10 presents the threats to validity.

8.2 The Experiment Design

This section presents the six steps designed to perform the experiment (See figure 8.1). These steps are explained in the next sub-sections.

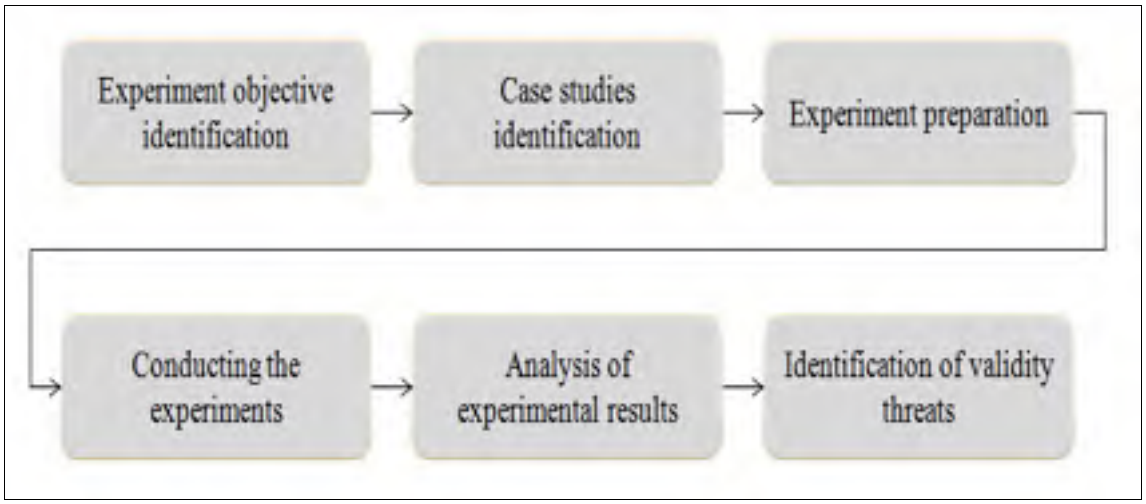

Figure 8.1 The experiment steps

8.2.1 Identification of Experiment Objective

The objective of the experimentation presented in this chapter is to verify the applicability of the proposed scaling factors framework. In this context, the term *"verification of applicability"* refers to the need to verify whether the concepts (i.e. meta-model and set of criteria) of the scaling factors framework can be applied to identify and rank the levels of granularity of sets of functional requirements specifications from various case studies.

In Chapter 4, the twelve (12) participants who participated in the experiment have applied the *E&Q* COSMIC technique (Conte, Iorio et Santillo, 2004) in order to classify a set of functional requirements specifications in accordance to their level of granularity into the

different *E&Q* COSMIC functional components. The experiment described in chapter 4 was conducted without following or using a well-detailed set of guidelines as proposed in the scaling factors framework to take into account the two (2) preliminary steps recommended in (COSMIC, 2007) and designed within the scaling factors framework:

- identification of the levels of granularity of requirements specifications; and
- identification and usage of size scaling factors.

The results of the experiment presented in Chapter 4 showed that the participants have made a high-level of miss-identification of the level of granularity of the requirements specifications, which affected the classification of those requirements specifications into the different *E&Q* COSMIC functional components: this resulted in an inaccurate approximation of the functional size of the software system under study, even though eight (8) out of the twelve (12) participants in the experiment had an average of twelve (12) years of experience in the software engineering industry at the time of the experiment.

This chapter will present the application of the two (2) preliminary steps recommended in (COSMIC, 2007) using the concepts of the scaling factors framework to identify the levels of granularity of the functional specifications of the four (4) case studies and the application of the interval scaling factors framework to approximate the functional size of these functional requirements specifications.

8.3 The Case Studies

8.3.1 uObserve software specification

This case study from (Trudel et Lavoie, 2008) presents the software requirements specifications of the first release of the “uObserve” system. It is worth mentioning that this case study was used in the experiment presented in Chapter 4. The uObserve system was developed for LESIA, an advanced interface and synthetic environment laboratory, at École de technologie supérieure (ÉTS) in Montreal, Canada. The system is meant as a proof of

concept for usability testing. This uObserve system consists of a sniffer library, a video camera and a microphone to record audio and video data of a user running a spied application while automatically synchronizing a log of events sent from the sniffer (records of keyboard and mouse movements).

The uObserve system is composed of two (2) subsystems: the uSleuth server, running on a different workstation from the uSpy client. uSpy is a sniffer library that would be integrated to an existing Java client application to be observed. uSpy sends a log of keyboard and mouse events to uSleuth via a Local Area Network (LAN) to uSleuth in order to record and play audio/video files that have been captured with the help of a camera with a microphone.

8.3.2 Banking information system

This second case study (INFSCI1024, 2011) presents the requirements specifications of a banking information system: this requirements specifications document was prepared by a group of undergraduate students course and then evaluated by their teacher at the University of Pittsburgh, U.S.A. 2011.

The requirements specifications document consists nine (9) pages divided into two (2) main sections, as follows:

- section 1 presents introductory information including background information, software purpose and scope, and software objectives; and
- section 2 presents use-case model including the use-case diagram and the use-case textual specifications that describe the functionality of the software to be developed.

8.3.3 The auction package system

This third case study (Stiefel, 2008) presents the requirements specifications of an on-line auction system where buyers and sellers can trade their products. This on-line auction system

attempts to encourage the buyers by giving them the equivalent of frequent flyer miles and foster the sense of community by providing a referral rewards program (Stiefel, 2008).

The requirements specifications document from (Stiefel, 2008) that presents this case study consists of ninety (90) pages divided into two (2) main sections, as follows:

- section 1 presents introductory information including background information, system purpose and scope, and system objectives; and
- section 2 presents use-case model including the use-case textual specifications of three (3) groups of use-cases: 'user registration', 'auction process workflow' and 'independent administrative' use-cases.

8.3.4 Automated teller machine system

This fourth case study (Bittner et Spence, 2003) presents the requirements specifications of an automated teller machine prepared by the authors of 'Use Case Modeling' book (Bittner et Spence, 2003). This case study consists of seventeen (17) pages of textual use-case specifications that present two (2) use-cases, namely 'Withdraw Cash' and 'Authenticate Customer'. The requirements specifications document that presents the case study is divided into two (2) main sections:

- section 1 presents introductory information including background information, software purpose and scope, and software objectives; and
- section 2 presents use-case model including the use-case diagram and the use-case textual specifications of 'withdraw cash' and 'authenticate user' use-cases.

8.4 Experiment preparation

The preparation of the experiment was done in two steps:

1) measurement of the functional size of the case studies; and
2) preparation of the case studies.

8.4.1 Measurement of the functional size of the case studies

The four (4) case studies used for experimentation in this chapter were presented in the previous section:

1. uObserve software system from (Trudel et Lavoie, 2008);
2. Banking information system from (INFSCI1024, 2011);
3. Auction Package (TAP) system from (Stiefel, 2008); and
4. Automated teller machine (ATM) system from (Bittner et Spence, 2003).

The functional size of the uObserve case study had already been measured (See table 4.4 in Chapter 4). On the other hand, the functional sizes of the three (3) other case studies were not available. In the first step of this experiment, the functional sizes for case studies 2 to 4 were measured by an expert measurer (i.e. Expert #B5).

Table 8.1 presents the functional sizes of the case studies 2 to 4 and the total effort (in minutes) expended to measure the functional size of these case studies. The measured functional sizes will be used as the reference values for determining the accuracy of the functional size approximation that will be calculated using the interval scaling factors.

Table 8.1 Measured functional size of case studies 2 to 4

Case study title	Measured functional size (in CFP)	Effort (in minutes)
Banking information system	96	104
Auction package system	393	255
Automated teller machine system	119	112

8.4.2 Preparation of the case studies

Prior to the usage of the four (4) case studies in experimentation, the specifications of the functional requirements of these case studies were modified by the principal researcher since these four (4) case studies were considered to be detailed and complete by (Trudel, 2012) and

(Expert #B5) and allowed to be measured accurately using a standardized functional size measurement method like COSMIC (ISO19761, 2011). Therefore, for the purpose of our experimentation for approximation of sizing these case studies could not be used 'as is'. Details about these modifications of the functional specifications will be presented next in the "Experimentation" section of each case study.

8.5 Experimentation with uObserve software system

8.5.1 Case study preparation

In this experiment, a modified version of the original uObserve requirements specifications document (Trudel et Lavoie, 2008) was prepared by the principal researcher. The modified version of the software requirements specifications document consists of:

- six (6) use-cases were kept ‘as is’ (i.e. without any modification of their specifications);
- four (4) use-cases were partially modified by removing portions of the specifications; and
- five (5) use-cases were completely modified by removing use-case specifications entirely.

8.5.2 Applying the scaling factors framework

By applying the scaling factors framework, fifteen (15) use-cases were identified at the use-case level:

- five (5) use-cases were assigned *'<unspecified> use-case'* as a scaling factor; and
- ten (10) use-case were assigned *'<specified> use-case'* as a scaling factor.

At the scenario level, the ten (10) use-cases that were assigned *'<specified> use-case'* as a scaling factor contained:

- six (6) scenarios were assigned '*<fully-specified> scenario'* as a scaling factor; and
- four (4) scenarios assigned '*<partially-specified> scenario*' as a scaling factor.

Table 8.2 presents the software requirements specifications (i.e. use-case specifications) of the uObserve software system along with the scaling factors assignment.

Table 8.2 Levels of granularity of the uObserve requirements specifications

Use Case Title	Level of granularity
1. Record an experiment *Main flow*: recording of new experiment *Alternative flows*: N/A	<Specified> use-case <Fully-specified> scenario N/A
2. Start uSleuth *Main flow*: start uSleuth *Alternative flows*: no camera found	<Specified> use-case <Fully-specified> scenario <Unspecified> scenario
3. Connect to uSleuth *Main flow*: connect uSpy to uSleuth *Alternative flows*: N/A	<Specified> use-case <Partially-specified> scenario N/A
4. Synchronize clocks	<Unspecified> use-case
5. Send Events	<Unspecified> use-case
6. Open a recorded experiment *Main flow*: open recorded experiment *Alternative flows*: N/A	<Specified> use-case <Partially-specified> scenario N/A
7. Playback an experiment *Main flow*: play experiment *Alternative flows*: N/A	<Specified> use-case <Fully-specified> scenario N/A
8. Pause playing *Main flow*: pause experiment Alternative flows: N/A	<Specified> use-case <Fully-specified> scenario N/A
9. Stop playing *Main flow*: stop playing an experiment *Alternative flows*: N/A	<Specified> use-case <Fully-specified> scenario N/A
10. Seek from image	<Unspecified> use-case
11. Scroll/Select from the event list	<Unspecified> use-case
12. Close (eject) experiment files *Main flow*: close experiment *Alternative flows*: N/A	<Specified> use-case <Partially-specified> scenario N/A
13. Display “About uObserve”	<Unspecified> use-case
14. Close uSleuth *Main flow*: close uSleuth *Alternative flows*: N/A	<Specified> use-case <Fully-specified> scenario N/A
15. Disconnect uSpy from uSleuth *Main flow*: disconnect uSpy *Alternative flow*: N/A	<Specified> use-case <Partially-specified> scenario N/A

8.5.3 Assigning interval scaling factors

After that, the interval scaling factors of the scaling factors framework were applied adopting the same functional size approximation approach suggested in the study of (Vogelezang et Prins, 2007), that is the "Equal size classification" approach to calculate an approximation of the functional size of the uObserve software system (See table 8.3):

- eleven (11) use-cases are assigned in the "Small" size band;
- three (3) use-cases are assigned in the "Medium" size band; and
- one (1) use-case is assigned in the "Large" size band.

Table 8.3 Assignment of interval scaling factors for uObserve system

Use Case Title	**Interval scaling factor**	
	Size band	**Numerical assignment (in CFP)**
1. Record an experiment	Large	10.7
2. Start uSleuth	Small	4.8
3. Connect to uSleuth	Small	4.8
4. Synchronize clocks	Medium	7.7
5. Send Events	Small	4.8
6. Open a recorded experiment	Medium	7.7
7. Playback an experiment	Medium	7.7
8. Pause playing	Small	4.8
9. Stop playing	Small	4.8
10. Seek from image	Small	4.8
11. Scroll/Select from the event list	Small	4.8
12. Close experiment files	Small	4.8
13. Display "About uObserve"	Small	4.8
14. Close uSleuth	Small	4.8
15. Disconnect uSpy from uSleuth	Small	4.8
Total approximated functional size (in CFP)		**86.6**
MRE from the reference value of 79.3 CFP		**9.2%**

The classification of the fifteen (15) use-cases into the four (4) size bands in table 8.4 is carried out based on:

- for use-cases that are assigned *'<specified> use-case'* as a scaling factor: number of *'data movements'* in each use-case will determine the appropriate size band; and
- for use-cases that are assigned *'<unspecified> use-case'* as a scaling factor: comparison is conducted with the functional size of similar use-cases that are completely specified (i.e. their functional size can be measured using standard measurement rules).

For example, for the 'Record an experiment' use-case, the level of granularity of its main and only scenario is <*fully-specified*> and ten (10) *data movements* were identified in its documented functional specification. Therefore, this use-case is classified in the 'Large' size band. On the other hand, the 'Pause playing' use-case has been assigned the *'<unspecified> use-case* scaling factor to identify its level of granularity. By comparing the 'Pause playing' use-case with other completely specified use-cases like 'Close experiment files' in which the total number of identified *data movements* is five (5) *data movements*. Therefore, the 'Pause playing' use-case is classified in the 'Small' size band.

Next, the number of use-cases in each size band is multiplied by the corresponding average functional size value (i.e. interval scaling factor) to result in total approximated functional size of approximately eighty-seven (87) COSMIC Function Point.

The Magnitude of Relative Error (MRE) quality equation of (Gene et Charles, 1996) is used to calculate the accuracy of the functional size approximation. The calculated functional size is next compared with the reference value of 79.3 CFP for the un-modified case study (Trudel, 2012).

8.5.4 Comparison & findings

This step compares the functional size approximation calculated by applying in interval scaling factors (See table 8.3) with the reference value of 79.3 CFP (table 4.4) of the original requirements specifications document: the approximated functional size using interval scaling factors is 86.6 CFP with an MRE value of 9%. This result of functional size

approximation illustrates that the proposed scaling factors framework is applicable to the functional specifications of the uObserve system: the meta-model and the set of criteria of the framework were used to identify the levels of granularity of the functional specifications and the interval scaling factors of (Vogelezang et Prins, 2007) were used to rank the levels of granularity - in CFP of the COSMIC measurement method - to approximate of the functional size of the uObserve system.

8.6 Experimentation with the banking information system

8.6.1 Case study preparation

In this experiment, a modified version of the original requirements specifications (INFSCI1024, 2011) of the Banking information system was prepared by the principal researcher. The modified version of the requirements specifications document consists of:

- four (4) use-cases were kept 'as is' (i.e. without any modification of specifications); and
- six (6) use-cases were completely modified by removing use-cases specifications entirely.

8.6.2 Applying the scaling factors framework

By applying the scaling factors framework, ten (10) use-cases were identified at the use-case level:

- six (6) use-cases were assigned *'<unspecified> use-case'* as a scaling factor; and
- four (4) use-case were assigned *'<specified> use-case'* as a scaling factor.

Table 8.4 presents the software requirements specifications (i.e. use-case specifications) of the Banking information system along with the scaling factors assignment.

Table 8.4 Levels of granularity of the banking information system

Use Case Title	Level of granularity
1. Login *Main flow:* login user *Alternative flows:* N/A	<Specified> use-case <Fully-specified> scenario N/A

Table 8.4 Levels of granularity of the banking information system (Continued)

Use Case Title	Level of granularity
2. Open account *Main flow:* open client account *Alternative flows:* N/A	<Specified> use-case <Fully-specified> scenario N/A
3. Deposit fund *Main flow:* deposit fund *Alternative flows:* N/A	<Specified> use-case <Partially-specified> scenario N/A
4. Withdraw fund *Main flow:* withdraw fund *Alternative flows:* N/A	<Specified> use-case <Partially-specified> scenario N/A
5. Transfer fund	<Unspecified> use-case
6. View account history	<Unspecified> use-case
7. Setup bill payment	<Unspecified> use-case
8. Apply for loan	<Unspecified> use-case
9. Evaluate loan	<Unspecified> use-case
10. Make loan decision	<Unspecified> use-case

At the scenario level, each use-case of the four (4) use-cases that were assigned *'<specified> use-case'* as a scaling factor contained one (1) scenario. Two (2) use-cases (i.e. use-case 1 and 2) are completely specified: they are assigned a scaling factor *<specified> use-case* at the use-case level, and their associated scenarios are assigned *<fully-specified> scenario*. Whereas, use-cases (3 and 4) are assigned scaling factor *<specified> use-case* at the use-case level and scaling factor *<partially-specified> scenario* at the use-case scenario level. The remaining six (6) use-cases are assigned a scaling factor *<unspecified>use-case* to indicate their level of granularity.

8.6.3 Assigning interval scaling factors

Table 8.5 presents the interval scaling factors assignment and the functional size approximation of the use-cases of the banking information system:

- three (3) use-case scenarios are assigned in the “Small” size band;
- two (2) use-cases are assigned in the “Medium” size band;

- two (2) use-cases are assigned in the “Large” size band; and
- three (3) use-cases are assigned in the “ Very Large” size band.

Table 8.5 Assignment of interval scaling factors for the banking information system

Use Case Title	Interval scaling factor	
	Size band	Numerical assignment (in CFP)
1. Login	Small	4.8
2. Open account	Medium	7.7
3. Deposit fund	Small	4.8
4. Withdraw fund	Small	4.8
5. Transfer fund	Large	10.7
6. View account history	Medium	7.7
7. Setup bill payment	V. Large	16.4
8. Apply for loan	V. Large	16.4
9. Evaluate loan	Large	10.7
10. Make loan decision	V. Large	16.4
Total approximated functional size (in CFP)		**100.4**
MRE from the reference value of 96 CFP		**3.5%**

The classification of the ten (10) use-cases into the four (4) size bands is carried out based on:

- for use-cases that are assigned *'<specified> use-case'* as a scaling factor: the number of *'data movements'* in each use-case will determine the appropriate size band; and
- for use-cases that are assigned *'<unspecified> use-case'* as a scaling factor: comparison is conducted with the functional size of similar use-cases that are completely specified (i.e. their functional size can be measured using standard measurement rules).

For example: for the 'Open an account' use-case, the level of granularity of its main and only scenario is <*fully-specified*>: seven (7) *data movements* were identified, and therefore this use-case is classified in the 'Medium' size band. On the other hand, the 'View account history' use-case has been assigned *'<unspecified> use-case* to identify its level of granularity. By comparing the 'View account history' use-case with other completely specified use-cases like

'Open an account' in which the total number of identified *data movements* is seven (7) *data movements*. Therefore, the 'View account history' use-case is also classified in the 'Medium' size band.

8.6.4 Comparison & findings

This step compares the functional size approximation calculated by applying in interval scaling factors (See table 8.5) with the reference value of 96 CFP (See table 8.1) of the original requirements specifications document: the approximated functional size using interval scaling factors is 100.4 CFP with an MRE value of 3.5%. This result of functional size approximation illustrates that the proposed scaling factors framework is applicable to the functional specifications of the banking information system: the meta-model and the set of criteria of the framework were used to identify the levels of granularity of the functional specifications and the interval scaling factors of (Vogelezang et Prins, 2007) were used to rank the levels of granularity - in CFP of the COSMIC measurement method - to approximate of the functional size of the banking information system.

8.7 Experimentation with auction package system

8.7.1 Case study preparation

In this experiment, a modified version of the original requirements specifications document of the Auction Package system (Stiefel, 2008) was prepared by the principal researcher. The modified version of the requirements specifications document consists of:

- twenty-one (21) use-cases were kept ‘as is’ (i.e. no modification of specifications); and
- sixteen (16) use-cases were completely modified by removing their specifications entirely.

8.7.2 Applying the scaling factors framework

By applying the scaling factors framework, thirty-seven (37) use-cases were identified at the use-case level:

- sixteen (16) use-cases were assigned *'<unspecified> use-case'* as a scaling factor; and
- twenty (21) use-cases were assigned *'<specified> use-case'* as a scaling factor.

Table 8.6 presents the use-cases of the on-line auction system along with the scaling factors assignment for each use-case (scenario) of the auction package system.

Table 8.6 Levels of granularity of the auction system

Use Case Title	Level of granularity
3.1 User Registration	
3.1.1 Register as new individual user *Main flow*	<Specified> use-case <Fully-specified> scenario
3.1.2 Register as new business user	<Unspecified> use-case
3.1.3 Response to 'confirm email' *Main flow*	<Specified> use-case <Fully-specified> scenario
3.1.4 TAP sends promotion email *Main flow*	<Specified> use-case <Partially-specified> scenario
3.1.5 User sends promotion email	<Unspecified> use-case
3.2 Auction process workflow	
3.2.1 Seller offers item to be auctioned *Main flow*	<Specified> use-case <Fully-specified> scenario
3.2.2 Seller withdraw item from auction	<Unspecified> use-case
3.2.3 Auction bid interval ends without bids	<Unspecified> use-case
3.2.4 Bidder places a proxy bid *Main flow*	<Specified> use-case <Fully-specified> scenario
3.2.5 Auction bid interval ends with one acceptable bid	<Unspecified> use-case
3.2.6 Buyer pays via electronic check *Main flow*	<Specified> use-case <Fully-specified> scenario
3.2.7 Buyer pays via snail mail check	<Unspecified> use-case

Table 8.6 Levels of granularity of the auction system (Continued)

Use Case Title	Level of granularity
3.2.8 24-hours buyer payment interval ends *Main flow*	<Specified> use-case <Partially-specified> scenario
3.2.9 Buyer pays via credit card or online agent	<Unspecified> use-case
3.2.10 During 24-hours interval, buyer pays via credit card or online agent *Main flow*	<Specified> use-case <Fully-specified> scenario
3.2.11 During 24-hours interval, buyer pays via electronic check *Main flow*	<Specified> use-case <Fully-specified> scenario
3.2.12 During 24-hours interval, buyer pays via snail mail check	<Unspecified> use-case
3.2.13 Administrator receives check from buyer *Main flow*	<Specified> use-case <Fully-specified> scenario
3.2.14 TAP does not receive buyer's check in seven days *Main flow*	<Specified> use-case <Fully-specified> scenario
3.2.15 Check bounces	<Unspecified> use-case
3.2.16 Check clears, and item shipped *Main flow*	<Specified> use-case <Fully-specified> scenario
3.2.17 192-hour shipping interval elapses *Main flow*	<Specified> use-case <Partially-specified> scenario
3.2.18 Buyer notifies TAP item received in good condition	<Unspecified> use-case
3.2.19 Buyer notifies TAP item received in bad condition *Main flow*	<Specified> use-case <Fully-specified> scenario
3.2.20 Buyer notifies TAP item received in good condition within confirmation interval *Main flow*	<Specified> use-case <Fully-specified> scenario
3.2.21 Confirmation period elapses without response from buyer	<Unspecified> use-case
3.2.22 Buyer notifies TAP item received in bad condition within confirmation interval *Main flow*	<Specified> use-case <Fully-specified> scenario

Table 8.6 Levels of granularity of the auction system (Continued)

Use Case Title	Level of granularity
3.2.23 Dispute resolved, and seller is scheduled for next weekly payment *Main flow*	<Specified> use-case <Fully-specified> scenario
3.2.24 Dispute resolved, and buyer is scheduled for next weekly payment	<Unspecified> use-case
3.2.25 Weekly payment process *Main flow*	<Specified> use-case <Fully-specified> scenario
3.3 Independent administrative functionality	
3.3.1 Delete user *Main flow*	<Specified> use-case <Fully-specified> scenario
3.3.2 Change parameters to computer awards	<Unspecified> use-case
3.3.3 Report TAP token balance *Main flow*	<Specified> use-case <Partially-specified> scenario
3.3.4 Withdraw TAP tokens	<Unspecified> use-case
3.3.5 Produce report of transactions *Main flow*	<Specified> use-case <Partially-specified> scenario
3.3.6 Edit generated email messages	<Unspecified> use-case
3.3.7 Report TAP token activity	<Unspecified> use-case

At the scenario level, the twenty-one (21) use-cases that were assigned *'<specified> use-case'* as a scaling factor contained:

- sixteen (16) scenarios were assigned '*<fully-specified> scenario'* as a scaling factor; and
- five (5) scenarios assigned '*<partially-specified> scenario*' as a scaling factor.

8.7.3 Assigning interval scaling factors

Table 8.7 presents the interval scaling factors assignment and the functional size approximation of the use-case scenarios of the Auction Package system:

- eleven (11) use-cases are assigned in the “Small” size band;
- seven (7) use-cases are assigned in the “Medium” size band;
- five (5) use-cases are assigned in the “Large” size band; and
- fourteen (14) use-cases are assigned in the “ Very Large” size band.

The classification of the thirty-seven (37) scenarios into the four (4) size bands is carried out based on:

- for scenarios that are assigned *'<fully-specified> or <partially-specified> scenario'* as a scaling factor: number of *'data movements'* in each scenario will determine the appropriate size band; and
- for scenarios that are assigned *'<unspecified> scenario'* as a scaling factor: a comparison is conducted with the functional size of similar scenarios that are completely specified (i.e. their functional size can be measured using standard measurement rules).

Table 8.7 Assignment of interval scaling factors for the auction system

Use Case Title	Interval scaling factor	
	Size band	Numerical assignment (in CFP)
3.1 User Registration		
3.1.1 Register as new individual user	Small	4.8
3.1.2 Register as new business user	Small	4.8
3.1.3 Response to 'confirm email'	Large	10.7
3.1.4 TAP sends promotion email	Small	4.8
3.1.5 User sends promotion email	Small	4.8
3.2 Auction Process Workflow		
3.2.1 Seller offers item to be auctioned	Large	10.7
3.2.2 Seller withdraw item from auction	Medium	7.7
3.2.3 Auction bid interval ends without bids	Small	4.8
3.2.4 Bidder places a proxy bid	V. Large	16.4
3.2.5 Auction bid interval ends with one acceptable bid	Small	4.8
3.2.6 Buyer pays via electronic check	V. Large	16.4
3.2.7 Buyer pays via snail mail check	V. Large	16.4
3.2.8 24-hours buyer interval ends	Small	4.8
3.2.9 Buyer pays via credit card or online agent	V. Large	16.4
3.2.10 During 24-hours interval, buyer pays via credit card or online agent	V. Large	16.4

Table 8.7 Assignment of interval scaling factors for the auction system (Continued)

Use Case Title	Interval scaling factor	
	Size band	Numerical assignment (in CFP)
3.2.11 During 24-hours interval, buyer pays via electronic check	V. Large	16.4
3.2.12 During 24-hours interval, buyer pays via snail mail check	V. Large	16.4
3.2.13 Administrator receives check from buyer	V. Large	16.4
3.2.14 TAP does not receive buyer's check in seven days	V. Large	16.4
3.2.15 Check bounces	Medium	7.7
3.2.16 Check clears, and item shipped	Medium	7.7
3.2.17 192-hour shipping interval elapses	Small	4.8
3.2.18 Buyer notifies TAP item received in good condition	V. Large	16.4
3.2.19 Buyer notifies TAP item received in bad condition	Large	10.7
3.2.20 Buyer notifies TAP item received in good condition within confirmation interval	V. Large	16.4
3.2.21 Confirmation period elapses without response from buyer	Medium	7.7
3.2.22 Buyer notifies TAP item received in bad condition within confirmation interval	Large	10.7
3.2.23 Dispute resolved, and seller is scheduled for next weekly payment	Medium	7.7
3.2.24 Dispute resolved, and buyer is scheduled for next weekly payment	Medium	7.7
3.2.25 Weekly payment process	V. Large	16.4
3.3 Independent administrative functionality		
3.3.1 Delete user	V. Large	16.4
3.3.2 Change parameters to awards	Small	4.8
3.3.3 Report TAP token balance	Small	4.8
3.3.4 Withdraw TAP tokens	V. Large	16.4

Table 8.7 Assignment of interval scaling factors for the auction system (Continued)

Use Case Title	Interval scaling factor	
	Size band	Numerical assignment (in CFP)
3.3.5 Produce report of transactions	Medium	7.7
3.3.6 Edit generated email messages	Large	10.7
3.3.7 Report TAP token activity	Small	4.8
Total approximated functional size (in CFP)		**390**
MRE from the reference value of 393 CFP		**0.7%**

For example, the 'Register new individual user' use-case, the level of granularity of its main and only scenario is *<fully-specified>*: four (4) *data movements* were identified. Therefore, this use-case is classified in the 'Small' size band. On the other hand, the use-case 'Register new business user' has been assigned *'<unspecified> use-case* to identify its level of granularity. By comparing the 'Register new business user' use-case with other completely specified use-cases like 'Register new individual user' in which the total number of identified *data movements* is four (4) *data movements*: the 'Register new business user' use-case is classified in the 'Small' size band.

8.7.4 Comparison & findings

This step compares the functional size approximation calculated by applying in interval scaling factors (See table 8.7) with the reference value of 393 CFP (See table 8.1) of the original requirements specifications document: the approximated functional size using interval scaling factors is 390 CFP with an MRE value of 0.7%. This result of functional size approximation illustrates that scaling factors framework is applicable to the functional specifications of the auction package system: the meta-model and the set of criteria of the framework were used to identify the levels of granularity of the functional specifications and the interval scaling factors of (Vogelezang et Prins, 2007) were used to rank the levels of granularity - in CFP of the COSMIC measurement method - to approximate of the functional size of the auction package system.

8.8 Experimentation with automated teller machine system

8.8.1 Case study preparation

In this experiment, a modified version of the original requirements specifications document of the Automated Teller Machine (Bittner et Spence, 2003) was prepared by the principal researcher. The modified version of the requirements specifications document consists of:

- seven (7) scenarios in the 'withdraw cash' use-case were kept 'as is' (i.e. without any modification of specifications); and
- ten (10) scenarios in the 'withdraw cash' and 'authenticate user' use-cases were modified by removing their specifications entirely.

8.8.2 Applying the scaling factors framework

By applying the scaling factors framework, two (2) use-cases were identified at the use-case level and were assigned *'<specified> use-case'* as a scaling factor. Table 8.8 presents the software requirements specifications (i.e. use-case specifications) of the Automated Teller Machine system along with the scaling factors assignment.

Table 8.8 Levels of granularity of the ATM system

Use Case Title	Level of granularity
1. Withdraw Cash	
Main flow: 1.1 Withdraw cash money	<Fully-specified>scenario
Alternative flows	
1.2 Specialist withdrawal facilities	<Partially-specified> scenario
1.3 Card handling	<Partially-specified> scenario
1.4 Receipt handling	<Partially-specified> scenario
1.5 Error handling	
1.5.1 Handling authentication errors	<unspecified> scenario
1.5.2 Handling dispensing errors	<Fully-specified>scenario
1.5.3 Handling money left by customer	<Fully-specified>scenario

Table 8.8 Levels of granularity of the ATM system (Continued)

Use Case Title	Level of granularity
1.5.4 Handling running out of resources	<unspecified> scenario
1.5.5 Handling security breaches	<Fully-specified>scenario
1.5.6 Handling quitting the session	<unspecified> scenario
1.5.7 Handling customer non-response	<unspecified> scenario
1.5.8 Handling log failure	<unspecified> scenario
2. Authenticate Customer	
Main flow: 2.1 authenticate customer	<unspecified> scenario
Alternative flows	
2.2 Handling bank communications	<unspecified> scenario
2.3 Handling stolen bank card	<unspecified> scenario
2.4 Handling invalid card information	<unspecified> scenario
2.5 Handling correct PIN not entered	<unspecified> scenario

At the scenario level, each scenario of the two (2) use-cases was assigned a scaling factor based on its level of granularity:

- seven (7) scenarios assigned '<*unspecified*> *scenario*' as a scaling factor.
- three (3) scenarios assigned '<*partially-specified*> *scenario*' as a scaling factor; and
- seven (7) scenarios were assigned '<*fully-specified*> *scenario'* as a scaling factor.

8.8.3 Assigning interval scaling factors

Table 8.9 presents the interval scaling factors assignment and the functional size approximation of the use-case scenarios of the Automated Teller Machine system:

- eight (8) use-case scenarios are assigned in the “Small” size band;
- five (5) use-case scenarios are assigned in the “Medium” size band;
- two (2) use-case scenarios is assigned in the “Large” size band; and
- two (2) use-case scenarios is assigned in the “ Very Large” size band.

Table 8.9 Assignment of interval scaling factors for the automated-teller machine system

Use Case Title	Interval scaling factor	
	Size band	Numerical assignment (in CFP)
1. Withdraw Cash		
1.1 Withdraw cash money	Large	10.7
1.2 Specialist withdrawal facilities	Medium	7.7
1.3 Card handling	Large	10.7
1.4 Receipt handling	Medium	7.7
1.5 Error handling		
1.5.1 Handling authentication errors	Small	4.8
1.5.2 Handling dispensing errors	Medium	7.7
1.5.3 Handling money left by customer	Medium	7.7
1.5.4 Handling running out of resources	Small	4.8
1.5.5 Handling security breaches	V. Large	16.4
1.5.6 Handling quitting the session	Small	4.8
1.5.7 Handling customer non-response	Medium	7.7
1.5.8 Handling log failure	Small	4.8
2. Authenticate Customer		
2.1 authenticate customer	Small	4.8
2.2 Handling bank communications	V. Large	16.4
2.3 Handling stolen bank card	Small	4.8
2.4 Handling invalid card information	Small	4.8
2.5 Handling correct PIN not entered	Small	4.8
Total approximated functional size (in CFP)		**142**
MRE from the reference value of 119 CFP		**19%**

For example, the 'Handling dispensing errors', scenario: its level of granularity is <*fully-specified*>: seven (7) *data movements* were identified in its functional specification. Therefore, this scenario is classified in the 'Medium' size band. On the other hand, the scenario 'Handling log failure' has been assigned *'<unspecified> scenario'* to identify its level of granularity. By comparing 'Handling log failure' with other completely specified scenarios like 'Handling dispensing errors' in which the total number of identified *data movements* is seven (7) *data movements*. Then, the 'Handling log failure' scenario is classified in the 'Medium' size band.

8.8.4 Comparison & findings

This step compares the functional size approximation calculated by applying in interval scaling factors (See table 8.9) with the reference value of 119 CFP (See table 8.1) of the original requirements specifications document: the approximated functional size using interval scaling factors is 142 CFP with an MRE value of 19%. This result of functional size approximation illustrates that the proposed scaling factors framework is applicable to the functional specifications of the automated teller machine system: the meta-model and the set of criteria of the framework were used to identify the levels of granularity of the functional specifications and the interval scaling factors of (Vogelezang et Prins, 2007) were used to rank the levels of granularity in CFP of the COSMIC measurement method to approximate of the functional size of the automated teller machine system.

8.9 Summary of findings

Table 8.10 presents a summary of the functional size approximation of the four (4) case studies used in the verification of the applicability of the scaling factors framework:

- the 1st column presents the reference functional sizes measured using the original documents of the four (4) case studies;
- the 2nd column presents the approximated functional sizes calculated using the interval scaling factors of the modified documents of the four (4) case studies; and
- the 3rd column presents the MRE values calculated using the reference functional sizes in the 1st column and the approximated functional sizes calculated using the interval scaling factors in the 2nd column.

A comparison is presented next between the average MRE value calculated in table 4.10 and the MRE value calculated in table 8.10 for the 'uObserve software' case study only since the same case study is used in Chapter 4 and in Section 8.5.

When the MRE value of the uObserve software in the 3rd column of 9.2% is compared with the average MRE value of 1361% calculated in Chapter 4 (See table 4.10), the comparison is performed with the 'most-likely' value in table 4.10 and therefore this gives difference percentage value of 1352%.

The results presented in table 8.10 shows that the concepts of the scaling factors framework are applicable to identify the levels of granularity of the functional specifications and to approximate the functional size of the 4 case studies used for experimentation.

Table 8.10 Summary of functional size measurement/approximation of the 4 case studies

Case study title	**Reference size (in CFP) measured using standard COSMIC method (1)**	**Approximated size (in CFP) using interval scaling factors (2)**	**MRE of (2) using reference size in (1)**
uObserve system	79.3 CFP	86.6 CFP	9.2%
Banking information system	96 CFP	100.4 CFP	3.5%
Auction package system	393 CFP	390 CFP	0.7%
Automated Teller Machine	119 CFP	142 CFP	19%

8.10 Validity Threats

8.10.1 Construct validity threat

A construct validity threat is associated to the validity of the experimental settings to reflect the subject under study (i.e. the applicability of the scaling factors framework). The quality of the functional specifications of the original version of the 'uObserve software system' was inspected by different groups of inspectors (i.e. personnel that inspect requirements quality aimed to assure high-quality requirements) and measurers (Trudel, 2012). On the other hand,

the quality of the functional specifications of the three (3) remaining case studies had not been inspected by inspectors.

To mitigate this potential threat, the measurer (i.e. Expert #B5) who measured the functional size of the original versions of these three (3) case studies was asked to evaluate the overall quality ranking of the functional specification in these three (3) case studies (See table 8.11) using the quality criteria presented by (Desharnais, Kocaturk et Abran, 2011).

It is worth mentioning that (Desharnais, Kocaturk et Abran, 2011) proposed five (5) levels quality ranking (A → E) for the functional specifications of early versions of software requirements specifications documents in which the (A) quality ranking denotes high-quality specifications and (E) quality ranking denotes poor quality specifications (See Page #23 in Chapter 1 for more information of this quality criteria and its ranking).

Table 8.11 Overall quality of the functional specifications of case studies 2 to 4

Case study title	**Overall quality ranking**
Banking information system	B
Auction package system	A
Automated teller machine system	B

The quality ranking of the three (3) case studies in table 8.10 indicates that the functional specifications in these case studies are well-detailed to allow a standardized functional size measurement method like COSMIC (ISO19761, 2011) to be used to obtain an accurate measurement of its functional size.

8.10.2 Internal validity threat

An internal validity threat is associated to the validity of the experimental results to changes in the design of the experiment. The original versions of the four (4) case studies used in the verification of the applicability of the scaling factors framework were considered to be detailed and complete (Trudel, 2012) & (Expert #B5). Therefore, the case studies could not

be used 'as is' for the purpose of our experimentation for approximation of sizing. To mitigate this potential threat, portions of functional specifications where removed from the original versions of the four (4) case studies to 'simulate' the usage of non-detailed and non-complete software requirements specifications documents like the modified versions used for experimentation in this chapter.

The principal researcher has designed the scaling factors framework and conducted himself the experimentation (i.e. the identification of the levels of granularity and the assignment of interval scaling factors) with the four (4) case studies presented in this chapter and not with the help of an independent participant: this has an advantage that the learning curve of the principal researcher to learn the concepts of the scaling factors framework is minimal. On the other hand, this can be considered as a potential threat to validity. To mitigate the impact of this potential threat, the principal researcher has applied the scaling factors framework five (5) months after preparing the experimentation material (i.e. the modified case studies).

8.10.3 External validity threat

An external validity threat is associated to the validity to generalize the experimental results which are obtained by applying the scaling factors framework to the case studies outside the experimental settings. This chapter presented the verification of the applicability of the scaling factors framework with only four (4) case studies that are different in the software type they represent, business need and software functional size. On the other hand, the measured functional size of the original versions of the four (4) case studies ranges between 79.3 CFP and 393 CFP. According to the 2011 release of ISBSG (ISBSG, 2011), almost sixty-five (65%) percent of completed software projects submitted to ISBSG which are measured using the COSMIC measurement (ISO19761, 2011) have a functional size of less than 400 COSMIC Function Point. This shows that the four (4) case studies used in this chapter represents a large portion of software projects submitted to ISBSG (ISBSG, 2011).

The principal researcher is not an expert in applying a standardized functional size measurement method like the COSMIC measurement method (ISO19761, 2011). On the other hand, the principal researcher was able to calculate a functional size approximation with MRE 0.7 to 19 percent of the four (4) case studies using only the concepts proposed in the scaling factors framework (i.e. identify the correct levels of granularity of the functional specifications in the case studies) and the interval scaling factors presented in the study of (Vogelezang et Prins, 2007).

The functional specifications of the four (4) case studies were changed by the principal researcher and not by an independent expert is a potential threat to validity that may limit the generalization of the results. To mitigate this potential threat, the principal researcher has made the changes to the functional specifications five (5) months prior to the usage of these modified versions of the case studies in experimentation (i.e. applying the scaling factors framework including the interval scaling factors). Future work may include replicating the experiments by modifying the functional specifications of the four (4) case studies by an independent expert in order to improve the generalization of the results.

CHAPTER 9

FUNCTIONAL SIZE APPROXIMATION USING THE *E&Q* COSMIC TECHNIQUE WITH PARTIAL SUPPORT OF THE SCALING FACTORS FRAMEWORK

9.1 Introduction

This chapter presents now the approximation of the functional size using the *E&Q* COSMIC technique of the four (4) case studies used in experimentation in Chapter 8. The research objective is to illustrate the value added of using interval scaling factors of framework over using the statistical table of the *E&Q* COSMIC technique in approximate sizing.

The four (4) case studies used in experimentation of Chapter 8 are:

1) uObserve software system (Trudel et Lavoie, 2008);
2) banking information system (INFSCI1024, 2011);
3) auction package system (Stiefel, 2008); and
4) automated teller machine system (Bittner et Spence, 2003).

This chapter is organized as follows: Section 9.2 presents the identification of the experiment objectives. Section 9.3 to 9.6 presents the functional size approximation of the case studies using the *E&Q* COSMIC technique. Section 9.7 a summary of the findings. Section 9.8 presents the threats to validity.

9.2 Identification of Experiment Objectives

The objective of the experimentation in this chapter is to explore the value added of using the scaling factors framework including the interval scaling factors over using the statistical table of the *E&Q* COSMIC technique on the accuracy of the functional size approximation.

The term *'experimentation'* refers to the approximation of the functional size using the *E&Q* COSMIC technique (Conte, Iorio et Santillo, 2004) of the four (4) case studies after applying

the scaling factors framework to identify the levels of granularity (See Chapter 8) of the functional requirements specifications of these case studies. It also refers to the comparison of the of the functional size approximated using the interval scaling factors of (Vogelezang et Prins, 2007) in Chapter 8 and the functional size approximated using the *E&Q* COSMIC technique in this chapter.

In this chapter, it is expected that using the *E&Q* COSMIC technique after applying the scaling factors framework to identify the levels of granularity of the functional requirements specification of the four (4) case studies will allow for the classification of these requirements specifications into the appropriate classes of the *E&Q* COSMIC functional components and also expected to improve the approximation of the functional size compared to approximation results presented in Chapter 4 (i.e. without well-detailed set of guidelines as proposed in the scaling factors framework).

9.3 Experimentation with the uObserve system

9.3.1 *E&Q* functional size approximation

The levels of granularity of the use-cases of the modified uObserve software have been identified using the scaling factors framework (See Section 8.5).

In this experiment the statistical table of the *E&Q* COSMIC technique is used to classify the use-cases of the modified uObserve software into the different *E&Q* functional components. Table 9.1 presents:

- the *E&Q* classification; and
- the approximation of the functional size of the modified uObserve software.

Using the identified levels of granularity of the use-cases presented in Section 8.5:

- the use-cases that are assigned a scaling factor <*specified*> *use-case*: data movements are identified either in the use-case ‘main flow’ or the use-case ‘alternative flows’ to classify

these use-cases to the appropriate functional component class. For example, use-cases (1, 2, 7, 8, 9, and 14) are assigned a scaling factor <*specified*>*use-cases* and each use-case consists of one (1) scenario that is a <*fully-specified*> *scenario*. Also, use-cases (3, 6, 12, and 15) consist of one (1) scenario that is a <*partially-specified*> scenario; and

- the use-cases (4, 5, 10, 11, and 13) that are assigned a scaling factor <*unspecified*> *use-case* and for which there exists no detail to describe their functional specification: they are grouped and classified as one (1) Small General Process (GP), even though a Small General Process (GP) requires the identification of at least six (6) Functional Processes.

Table 9.1 *E&Q* classification and functional size approximation of the uObserve system

Use Case Title	***E&Q* classification**	**Functional size approximation (min, most-likely, max)**
1. Record an experiment	Large FP	(8, 10.5, 14)
2. Start uSleuth	Small FP	(2, 3.9, 5)
3. Connect to uSleuth	Medium FP	(5, 6.9, 8)
5. Open a recorded experiment	Medium FP	(5, 6.9, 8)
6. Playback an experiment	Medium FP	(5, 6.9, 8)
7. Pause playing	Small FP	(2, 3.9, 5)
8. Stop playing	Small FP	(2, 3.9, 5)
12. Close (eject) experiment files	Small FP	(2, 3.9, 5)
14. Close uSleuth	Small FP	(2, 3.9, 5)
15. Disconnect uSpy from uSleuth	Small FP	(2, 3.9, 5)
4. Synchronize clocks 5. Send Events 10. Seek from image 11. Scroll/Select from the event list 13. Display "About uObserve"	Small GP	(20, 60, 110)
Total approximated functional size (in CFP)		**(55, 115, 178)**
MRE from the reference value of 79.3 CFP		**(30%, 45%, 124%)**

The *E&Q* COSMIC technique (Conte, Iorio et Santillo, 2004) assigns a set of functional size values (minimum, most likely, and maximum) to each identified functional component: these

functional size values are arithmetically summed to provide the overall approximation result (minimum, most likely, and maximum).

9.3.2 Comparison & findings

This step compares the approximated functional size calculated (See table 9.1) using the statistical table of the *E&Q* COSMIC technique with the reference value (See table 4.4 in Chapter 4) of 79.3 CFP that represents the functional size measurement of the original requirements specifications document:

- when the approximated functional size of (Min: 55 CFP, Most-likely: 115 CFP, Max: 178 CFP) is compared to the reference value of 79.3 CFP, this gives a vector of MRE values of (Min: 30%, Most-likely: 45%, Max: 124%); and
- when the MRE value of 45% that is calculated based on the 'Most-likely' functional size of 115 CFP is compared to the MRE value of 9% that is calculated based on the approximated functional size value of 86.6 using interval scaling factors, this gives a difference percentage value of 36%.

The comparison results of the functional size approximation and the corresponding MRE values illustrates the improvement on the accuracy of the functional size when using the scaling factors framework including the interval scaling factors rather than the *E&Q* COSMIC technique. The Magnitude of Relative Error (MRE) calculated when using the interval scaling factors of the framework is 36% less than the calculated MRE when using the *E&Q* COSMIC technique in approximate sizing.

9.4 Experimentation with the banking information system

9.4.1 *E&Q* functional size approximation

The levels of granularity of the use-cases of the banking information system are identified using the scaling factors framework (See Section 8.6).

In this experiment the statistical table of the *E&Q* COSMIC technique is used to classify the use-cases of the modified banking information system into the different *E&Q* functional components. Table 9.2 presents:

- the *E&Q* classification; and
- the approximation of the functional size of the modified banking information system.

Using the identified levels of granularity of the use-cases presented in Section 8.6:

- the use-case cases that are completely specified: counting the *data movements* presents a form of standard measurement of their functional size in accordance to the COSMIC measurement method (ISO19761, 2011); and
- the use-cases that are not totally specified (i.e. their associated use-case scenarios are not assigned a scaling factor <*fully-specified*> *use-case scenario*) they were classified as a *'Small General Process'* (GP) to calculate an approximation of their functional size.

Table 9.2 *E&Q* classification and functional size approximation of the banking information system

Use Case Title	***E&Q* approximation**	**Functional size approximation (min, most-likely, max)**
1. Login	Medium FP	(5, 6.9, 8)
2. Open account	Medium FP	(5, 6.9, 8)
3. Deposit fund	Small GP	(2, 3.9, 5)
4. Withdraw fund	Medium FP	(5, 6.9, 8)
5. Transfer fund	Small GP	(20, 60, 110)
6. View account history		
7. Setup bill payment		
8. Apply for loan		
9. Evaluate loan		
10. Make loan decision		
Total approximated functional size (in CFP)		**(37, 85, 139)**
MRE from the reference value of 96 CFP		**(61%, 12.3%, 43%)**

9.4.2 Comparison & findings

This step compares the approximated functional size calculated (See table 9.2) using the statistical table of the *E&Q* COSMIC technique with the reference value (See table 8.1 in Chapter 8) of 96 CFP that represents the functional size measurement of the original requirements specifications document:

- when the approximated functional size of (Min: 37 CFP, Most-likely: 85 CFP, Max: 139 CFP) is compared to the reference value of 96 CFP, this gives a vector of MRE values of (Min: 61%, Most-likely: 12.3%, Max: 43%); and
- when the MRE value of 12.3% that is calculated based on the 'Most-likely' functional size of 85 CFP is compared to the MRE value of 3.5% that is calculated based on the approximated functional size value of 100.4 CFP using interval scaling factors, this gives a difference percentage value of 8.8%.

The comparison results of the functional size approximation and the corresponding MRE values illustrates the improvement on the accuracy of the functional size when using the scaling factors framework including the interval scaling factors rather than the *E&Q* COSMIC technique. The Magnitude of Relative Error (MRE) calculated when using the interval scaling factors of the framework is 8.8% less than the calculated MRE when using the *E&Q* COSMIC technique in approximate sizing.

9.5 Experimentation with auction package system

9.5.1 *E&Q* functional size approximation

The levels of granularity of the use-cases of the auction package system are identified using the scaling factors framework (See Section 8.7).

In this experiment the statistical table of the *E&Q* COSMIC technique is used to classify the use-cases of the modified auction package system into the different *E&Q* functional components. Table 9.3 presents:

- the *E&Q* classification; and
- the approximation of the functional size of the modified auction package system.

Using the identified levels of granularity of the use-cases presented in Section 8.7:

- the use-cases that are assigned a scaling factor <*specified*> *use-case* to identify their level of granularity are assigned in accordance to the identification of the data movements in their functional specifications; and
- the use-cases that are assigned a scaling factor <*unspecified*> *use-case* were grouped as one (1) 'Large General Process': and consists of sixteen (16) use-cases.

Table 9.3 *E&Q* classification and the functional size approximation of the auction system

Use Case Title	***E&Q* classification**	**Functional size approximation (min, most-likely, max)**
3.1.1 Register as new individual user	Small FP	(2, 3.9, 5)
3.1.3 Response to 'confirm email'	Large FP	(8, 10.5, 14)
3.1.4 TAP sends promotion email	Small FP	(2, 3.9, 5)
3.2.1 Seller offers item to be auctioned	Large FP	(8, 10.5, 14)
3.2.4 Bidder places a proxy bid	V. Large FP	(2, 3.9, 5)
3.2.6 Buyer pays via electronic check	V. Large FP	(14, 23.7, 30)
3.2.8 24-hours buyer payment interval ends	Small FP	(2, 3.9, 5)
3.2.10 During 24-hours interval, buyer pays via credit card or online agent	V. Large FP	(14, 23.7, 30)
3.2.11 During 24-hours interval, buyer pays via electronic check	V. Large FP	(14, 23.7, 30)
3.2.13 Administrator receives check from buyer	V. Large FP	(14, 23.7, 30)
3.2.14 TAP does not receive buyer's check in seven days	Large FP	(8, 10.5, 14)
3.2.16 Check clears, and item shipped	Medium FP	(5, 6.9, 8)
3.2.17 192-hour shipping interval elapses	Small FP	(2, 3.9, 5)

Table 9.3 *E&Q* classification and the functional size approximation of the auction system (Continued)

Use Case Title	E&Q classification	Functional size approximation (min, most-likely, max)
3.1.2 Register as new business user	Large GP	(60, 130, 220)
3.1.5 User sends promotion email		
3.2.2 Seller withdraw item from auction		
3.2.3 Auction bid interval ends without bids		
3.2.5 Auction bid interval ends with one acceptable bid		
3.2.7 Buyer pays via snail mail check		
3.2.9 Buyer pays via credit card or online agent		
3.2.12 During 24-hours interval, buyer pays via snail mail check		
3.2.15 Check bounces		
3.2.18 Buyer notifies TAP item received in good condition		
3.2.21 Confirmation period elapses without response from buyer		
3.2.24 Dispute resolved, and buyer is scheduled for next weekly payment		
3.3.2 Change parameters to computer awards		
3.3.4 Withdraw TAP tokens		
3.3.6 Edit generated email messages		
3.3.7 Report TAP token activity		
3.2.19 Buyer notifies TAP item received in bad condition	Large FP	(8, 10.5, 14)
3.2.20 Buyer notifies TAP item received in good condition within confirmation interval	Large FP	(8, 10.5, 14)
3.2.22 Buyer notifies TAP item received in bad condition within confirmation interval	Large FP	(8, 10.5, 14)
3.2.23 Dispute resolved, and seller is scheduled for next weekly payment	Medium FP	(5, 6.9, 8)

Table 9.3 *E&Q* classification and the functional size approximation of the auction system (Continued)

Use Case Title	E&Q classification	Functional size approximation (min, most-likely, max)
3.2.25 Weekly payment process	V. Large FP	(14, 23.7, 30)
3.3.1 Delete user	V. Large FP	(14, 23.7, 30)
3.3.3 Report TAP token balance	Small FP	(2, 3.9, 5)
3.3.5 Produce report of transactions	Large FP	(8, 10.5, 14)
Total approximated functional size		**(234, 403, 569) CFP**
MRE from the reference value of 393 CFP		**(40%, 2.5%, 44%)**

9.5.2 Comparison & findings

This step compares the approximated functional size calculated (See table 9.3) using the statistical table of the *E&Q* COSMIC technique with the reference value (See table 8.1 in Chapter 8) of 393 CFP that represents the functional size measurement of the original requirements specifications document:

- when the approximated functional size of (Min: 234 CFP, Most-likely: 403 CFP, Max: 569 CFP) is compared to the reference value of 393 CFP, this gives a vector of MRE values of (Min: 40%, Most-likely: 2.5%, Max: 44%); and
- when the MRE value of 2.5% that is calculated based on the 'Most-likely' functional size of 403 CFP is compared to the MRE value of 0.7% that is calculated based on the approximated functional size value of 390 CFP using interval scaling factors, this gives a difference percentage value of 1.8%.

The comparison results of the functional size approximation and the corresponding MRE values illustrates the improvement on the accuracy of the functional size when using the scaling factors framework including the interval scaling factors rather than the *E&Q* COSMIC technique. The Magnitude of Relative Error (MRE) calculated when using the interval scaling factors of the framework is 1.8% less than the calculated MRE when using the *E&Q* COSMIC technique in approximate sizing.

9.6 Experimentation with automated teller machine system

9.6.1 *E&Q* functional size approximation

The levels of granularity of the use-cases of the automated teller machine system are identified using the scaling factors framework (See Section 8.8).

In this experiment the statistical table of the *E&Q* COSMIC technique is used to classify the use-cases of the modified automated teller machine system into the different *E&Q* functional components. Table 9.4 presents:

- the *E&Q* classification; and
- the approximation of the functional size of the modified automated teller machine system.

Using the identified levels of granularity of the use-cases presented in Section 8.8:

- the scenarios that are assigned a scaling factor <*fully-specified*> or <*partially-specified*> *scenario* to identify their level of granularity are assigned in accordance to the identification of the data movements in their functional specifications; and
- the scenarios that are assigned a scaling factor <*unspecified*> *scenario* were grouped as one (1) 'Medium General Process' and consists of ten (10) scenarios.

Table 9.4 *E&Q* classification and functional size approximation of automated-teller machine system

Use Case Title	***E&Q* classification**	**Functional size approximation (min, most-likely, max)**
1.1 Withdraw cash money	Large FP	(8, 10.5, 14)
1.2 Specialist withdrawal facilities	Medium FP	(5, 6.9, 8)
1.3 Card handling	Large FP	(8, 10.5, 14)
1.4 Receipt handling	Large FP	(8, 10.5, 14)
1.5.2 Handling dispensing errors	Medium FP	(5, 6.9, 8)
1.5.3 Handling money left by customer	Medium FP	(5, 6.9, 8)
1.5.5 Handling security breaches	Very Large FP	(14, 23.7, 30)

Table 9.4 *E&Q* classification and functional size approximation of automated-teller machine system (Continued)

Use Case Title	***E&Q* classification**	**Functional size approximation (min, most-likely, max)**
1.5.1 Handling authentication errors	Medium GP	(40, 90, 160)
1.5.4 Handling out of resources		
1.5.6 Handling quitting the session		
1.5.7 Handling customer non-response		
1.5.8 Handling log failure		
2.1 authenticate customer		
2.2 Handling bank communications		
2.3 Handling stolen bank card		
2.4 Handling invalid card information		
2.5 Handling correct PIN not entered		
Total approximated functional size (in CFP)		**(93, 166, 256)**
MRE from the reference value of 119 CFP		**(21.8%, 39.4%, 115%)**

9.6.2 Comparison & findings

This step compares the approximated functional size calculated (See table 9.4) using the statistical table of the *E&Q* COSMIC technique with the reference value (See table 8.1 in Chapter 8) of 119 CFP that represents the functional size measurement of the original requirements specifications document:

- when the approximated functional size of (Min: 93 CFP, Most-likely: 166 CFP, Max: 256 CFP) is compared to the reference value of 119 CFP, this gives a vector of MRE values of (Min: 21.8%, Most-likely: 39.4%, Max: 115%); and
- when the MRE value of 39.4% that is calculated based on the 'Most-likely' functional size of 166 CFP is compared to the MRE value of 19% that is calculated based on the approximated functional size value of 142 CFP using interval scaling factors, this gives a difference percentage value of 20.4%.

The comparison results of the functional size approximation and the corresponding MRE values illustrates the improvement on the accuracy of the functional size when using the

scaling factors framework including the interval scaling factors rather than the *E&Q* COSMIC technique. The Magnitude of Relative Error (MRE) calculated when using the interval scaling factors of the framework is 20.4% less than the calculated MRE when using the *E&Q* COSMIC technique in approximate sizing.

9.7 Summary of Findings

Table 9.5 presents a summary of the functional size approximation of the four (4) case studies used in experimentation in this chapter:

- the 1st column presents the reference functional sizes measured using the original documents of the four (4) case studies;
- the 2nd column presents the approximated functional sizes calculated using the interval scaling factors of the modified documents of the four (4) case studies;
- the 3rd column presents the approximated functional sizes calculated using the *E&Q* COSMIC technique of the modified documents of the four (4) case studies;
- the 4th column presents the MRE values calculated using the reference functional sizes in the 1st column and the approximated functional sizes calculated using the interval scaling factors in the 2nd column;
- the 5th column presents the MRE values calculated using the reference functional sizes in the 1st column and the approximated functional sizes calculated using the *E&Q* COSMIC technique in the 3rd column; and
- the 6th column presents the arithmetic difference of MRE values presented in the 4th and the 5th column.

It is worth mentioning that approximated functional sizes in the 2nd and the 3rd columns were calculated after having the levels of granularity of the use-case case in the four (4) case studies identified using the scaling factors framework (see Section 8.5 to 8.8 in Chapter 8).

When the MRE values in the 4th and the 5th columns are compared for each case study (the comparison is performed with the 'most-likely' value in the 5th column), this gives difference percentage values range from 1.8 to 36%.

This gives an indication that using the scaling factors framework including the interval scaling factors to approximate the functional size of the four (4) case studies provides more accurate approximation than using the statistical table of the *E&Q* COSMIC technique.

Table 9.5 Summary of functional size measurement/approximation of the 4 case studies

Case study title	**Reference size measured using standard COSMIC method (1)**	**Approximated size using interval scaling factors (2)**	**Approximated size using the E&Q COSMIC technique (Min, Most-likely, Max) (3)**	**MRE of (2) using reference sizes in (1)**	**MRE of (3) using reference sizes in (1)**	**Difference of MRE values in the 4th & 5th column**
uObserve system	79.3 CFP	86.6 CFP	(55, 115, 178) CFP	9.2%	(30, 45, 124)%	36%
Banking information system	96 CFP	100.4 CFP	(37, 85, 139) CFP	3.5%	(61, 12.3, 43)%	8.8 %
Auction package system	393 CFP	390 CFP	(234, 403, 569) CFP	0.7%	(40, 2.5, 44)%	1.8%
Automated Teller Machine	119 CFP	142 CFP	(93, 166, 256) CFP	19%	(22, 39, 115)%	20.4%

9.8 Validity Threats

9.8.1 Construct validity threats

A construct validity threat is associated to the validity of the experimental settings to reflect the subject under study. The quality of the functional specifications of case studies 2 to 4 had not been inspected by inspectors. For more information on how this potential threat to validity is mitigated, the readers are referred to Section 8.10.1.

9.8.2 Internal validity threats

An internal validity threat is associated to the validity of the experimental results to changes in the design of the experiment. The following internal validity threats have been identified:

- the original versions of the four (4) case studies were considered to be detailed and complete by (Trudel, 2012) & (Expert #B5). Therefore, the case studies could not be used 'as is' for the purpose of our experimentation for approximation of sizing;
- the principal researcher has designed the scaling factors framework and conducted himself the approximation of the functional size; and

For more information on how these potential threats to validity are mitigated, the readers are referred to Section 8.10.2.

Another threat to validity is the statistical table of the *E&Q* COSMIC technique which was used 'as is' in the approximation of the functional size without any check on the validity of its contents: the assessment of the ranges introduced in this statistical table was out of the scope of this research project.

9.8.3 External validity threats

An external validity threat is associated to the validity to generalize the experimental results obtained from approximating the functional size of the case studies using the *E&Q* COSMIC technique. The following external validity threats have been identified:

- this chapter presented the approximation of the functional size using the *E&Q* COSMIC technique with only four (4) case studies; and
- the functional specifications of the four (4) case studies were modified by the principal researcher and not by an independent expert.

For more information on how these potential threats to validity are mitigated, the readers are referred to Section 8.10.3.

Another potential validity threat is the principal researcher is not an expert in applying the *E&Q* COSMIC technique. On the other hand, the principal researcher has only used a subset of the *E&Q* COSMIC technique (i.e. the statistical table) to calculate an approximation of the functional size: this is a very simple step, which did not require the use of expert opinion and subjective judgment. In other words, the principal researcher has replaced the subjective step of the E&Q COSMIC technique for the classification by the 1st recommended step in (COSMIC, 2007) through the use of the concepts of the scaling factors framework proposed in this research and the 2nd recommended step in (COSMIC, 2007) using the statistical table of the *E&Q* COSMIC technique.

CONCLUSION

The goal of this research project was to improve one of the inputs of the a priori effort estimation process, and in particular the functional size approximation of software development projects. The main objective of this research project was to design a framework to assign scaling factors for identifying the level of granularity of functional requirements specifications of software, which are typically documented at different levels of granularity at early stages of the software development life cycle.

To achieve this research goal, the following specific research objectives were specified:

1. an investigation of the impact of the scaling factors issue on the accuracy of the results of the functional size approximation;
2. a proposed new framework to assign scaling factors to early functional requirements specifications to identify and rank their levels of granularity:
   (a) a set relevant concepts and their relationships that need to be collected by the requirements engineers to allow full functional specification of software functional requirements;
   (b) a set of scaling factors to rank the levels of granularity of functional requirements specifications;
   (c) a meta-model that captures the relevant concepts with their relationships and the defined scaling factors; and
   (d) a set of criteria that identify the levels of granularity of functional requirements specifications.
3. the proposed scaling factors framework verified for usability and applicability.
4. an investigation of the impact of using the framework on the accuracy when the functional size is approximated using the statistical table of the E&Q COSMIC technique.

The research objectives were achieved using several software engineering standards, models, and frameworks, including:

- (ISO19761, 2011): the COSMIC International standard for software functional size measurement;
- (COSMIC, 2011): Guideline for assuring the accuracy of measurements v. 3.0.1;
- (Desharnais, Kocaturk et Abran, 2011): using the COSMIC method to evaluate the quality of the documentation of Agile user stories;
- (Vogelezang et Prins, 2007): Approximate size measurement with the COSMIC method: factors of influence;
- (COSMIC, 2007): Advanced & Related Topics v 3.0;
- (Azzouz et Abran, 2004): the proposal for functional size approximation of the requirements specifications documented using RUP levels of granularity;
- (IEEE, 1998): IEEE-830 Recommended practice for software requirements specifications;
- research studies (2000 - 2012) on software functional size measurement and requirements analysis and specification.

Defining the various concepts needed to perform a standard functional size measurement using the COSMIC standard, and the experimental studies conducted by (Trudel, 2012) have helped us to produce a rigorous and comprehensive research methodology. It also allowed us to focus on the design of the experimental studies in accordance to the best practices in the empirical software engineering field. The research work proposed by (Desharnais, Kocaturk et Abran, 2011) of evaluating the quality of the documentation of agile user stories using the COSMIC international standard was a valuable contribution to observe and analyze the research issue faced by an independent researcher in the software measurement field.

Research Contributions

The research contributions of this research project are classified into following categories and are explicitly related to each specific research objective:

1. **Document objectively and quantitatively the degree of reproducibility and accuracy the of an existing functional size approximation technique:** an experimental study was conducted with twelve (12) practitioners to approximate the functional size of the

uObserve software specifications using only the rules and concepts of the *E&Q* COSMIC technique:

- eight (8) of the 12 practitioners had an average experience of 12 years but misidentified the number of functional components in the case study by an average of 74%;
- the other four (4) practitioners had limited experience but misidentified the number of functional components in the case study by an average of 17%;

The functional size approximation using only the rules and the concepts of the *E&Q* COSMIC technique did not lead to reproducible results by the twelve (12) participants conducted this experiment, and resulted of an average difference of (Min: +158%, Most-likely: +87.4%, Max: +74%).

Further, the 12 practitioners have misclassified the identified functional components which resulted of an average magnitude of relative error of (Min: 684%, Most-likely: 1502%, Max: 2546%) of the approximated functional size.

It is worth mentioning that after fifteen years of the initial publication of the *Early & Quick* technique (Meli, 1997) and eight years after the publication of the COSMIC variant (Conte, Iorio et Santillo, 2004) no research to date have investigated whether or not this technique was meeting its stated objective, and what factors were an important source of errors. This research contribution corresponds to research objective #1.

2. **A novel framework to assign scaling factors to software functional requirements:** this included designing a novel framework that objectively re-assign the responsibility of identifying the levels of granularity of functional requirements specifications to the requirements engineers who elicit software requirements from the stockholders of software projects to assign such elicited requirements specifications scaling factors to identify and rank their level of granularity. This responsibility is currently handled by the personnel involved in the approximation process of software functional size who subjectively use intuition, experience or opinion of the field experts to identify the levels

of granularity of functional requirements specifications. This research contribution corresponds to research objective #2.

3. **Verification of the usability of the scaling factors framework:** the usability of the scaling factors framework was verified by conducting three (3) experimental sessions with the same case study involving three (3) different groups of industry practitioners who have different expertise profile in which most of the practitioners are involved in different stages of the software development life cycle:
   - all the participants in three (3) experimental sessions were able to identify correct number of functional components of the case study (i.e. the use-cases and their scenarios);
   - sixty-one (61%) percent of the participants in the 3 sessions identified the correct levels of granularity of the functional components;
   - fifteen (15%) percent of participants made one (1) mistake in identifying the levels of granularity of the functional components;
   - seven (7%) percent of participants made two (2) mistakes in identifying the levels of granularity of the functional components;
   - five (5%) percent of participants made three (3) mistakes in identifying the levels of granularity of the functional components;
   - ten (10%) percent of participants made four (4) mistakes in identifying the levels of granularity of the functional components; and
   - two (2%) percent of participants made six (6) mistakes in identifying the levels of granularity; of the functional components.

   This research contribution corresponds to research objective #3.

It is recommended that researchers and the users of such approximation methods to take into consideration the two preliminary steps recommended in (COSMIC, 2007) prior to the approximate of software functional size:

- identification of the levels of granularity of the functional requirements specifications;
- identification and usage of size scaling factors.

4. **Verification of the applicability of the scaling factors framework:** the verification of the applicability of the scaling factors framework was verified by applying the concepts of the scaling factors framework to 4 case studies which represents different software applications that are different in software type, business objective, context, and software functional size. It also included the approximation of the case studies functional size using interval scaling factors presented in the study of (Vogelezang et Prins, 2007):
   - the MRE of the approximated functional size of 86.6 CFP of the uObserve software compared to the reference measured functional size of 79.3 is equal to 9.2%;
   - the MRE of the approximated functional size of 100.4 CFP of the banking information system compared to the reference measured functional size of 96 CFP is equal to 3.5%;
   - the MRE of the approximated functional size of 390 CFP of the auction package system compared to the reference measured functional size of 393 CFP is equal to 0.7%; and
   - the MRE of the approximated functional size of 142 CFP of the automated teller machine system compared to the reference measured functional size of 119 CFP is equal to 19%.

   This research contribution corresponds to research objective #3.

5. **Document objectively and quantitatively the degree of accuracy the *E&Q* COSMIC approximation with the partial support of the scaling factors framework:** the approximation of the functional size using the statistical table of the *E&Q* COSMIC technique is calculated after identifying the levels of granularity of the functional specifications of the four (4) case studies using the scaling factors framework: these approximations of the functional size are next compared with the approximations calculated using interval scaling factors of the framework:
   - the MRE of the approximated 'most-likely' functional size of 115 CFP of the uObserve software compared to the reference measured functional size of 79.3 is equal to 45%;

- the MRE of the approximated 'most-likely' functional size of 85 CFP of the banking information system compared to the reference measured functional size of 96 CFP is equal to 12.3%;
- the MRE of the approximated 'most-likely' functional size of 403 CFP of the auction package system compared to the reference measured functional size of 393 CFP is equal to 2.5%; and
- the MRE of the approximated 'most-likely' functional size of 142 CFP of the automated teller machine system compared to the reference measured functional size of 119 CFP is equal to 39%.

When the MRE values in the 4th and the 5th research contributions are compared for each case study (the comparison is performed with the 'most-likely' value in the 5th research contribution), this gives difference percentage values range from 1.8 to 36%: this gives an indication that using the scaling factors framework including the interval scaling factors to approximate the functional size of the four (4) case studies provides more accurate approximation than using the statistical table of the E&Q COSMIC technique. This research contribution corresponds to research objective #4.

Research Users

The users of this research project who will benefit from the main research outcome (i.e. the scaling factors framework):

- requirements engineers: the scaling factors framework will help the requirements engineers in collecting and documenting all the relevant information needed by the software measurers, and also it will help them ranking the levels of granularity of the software requirements specifications;
- measurers of software functional size: the scaling factors framework will help the measurers by providing objective information about the levels of granularity of the functional requirements specifications to calculate an improved approximation of software functional size: such improvement in the approximation of the functional size will help the

estimators of the effort of software projects to build more accurate effort estimation models of software development projects;

- software requirements quality assurance personnel: the scaling factors framework will provide early indicators about the quality of the software requirements specifications to the quality assurance personnel who will be able to report quality improvement recommendations to the requirements engineers who documented those specifications;
- software measurement research community: develop new and improved methodologies that assign scaling factors to early versions of software requirements specifications to identify the level of granularity at early phases of the software development life cycle; and
- software development organizations: open the door to design new tools to calculate more reasonable approximation of software functional size.

Future Work

The research presented in this thesis can lead to further work to enhance the understanding of the scaling factors issue in both academia and industry. In this thesis, several software engineering standards and models were investigated in order to improve the early approximation of the functional size of the functional requirements specifications of software development projects. Accordingly, the following future work can be pursued based on the results and the methodologies used in this thesis:

- **software measurement research community:** this thesis highlights to the software measurement research community the importance of the scaling factors issue and its impact on the software development life cycle and in particular its impact on the early approximation of the software functional size. In the literature, few researchers have recognized the quality issue of the 'completeness' of the documentation of the functional requirements specifications and its impact on the early approximation of the software functional size.

  This thesis opens the door to improve the verification of the scaling factors framework by conducting more experimental studies with new groups of practitioners in software engineering field, and with different requirements specifications documents of software

applications of different types. Further, this thesis opens the door to software measurement research community to propose new research methodologies to tackle the scaling factors research issue.

- **software development organizations:** the software functional size is one of the primary inputs for the software effort estimation process: in this process the managers and the technical leaders of software projects use the available methods and techniques in order to build effort estimation models that present the expected effort and duration of such software projects which helps the projects managers and the technical leaders in arranging the contractual agreements with their clients and in resources acquisition and distribution during the whole life cycle of those software development projects.
  This thesis highlighted the impact of the scaling factors issue on the early approximation of the software functional size which typically affects the effort estimation process and therefore affects success of each phase of the development life cycle of software projects. This thesis opens the door for the software development organizations to tailor the scaling factors framework to their development life cycle and in particular prior to the functional size approximation process.

- **domain extension:** the functional user requirements are one of the primary inputs for the research project presented in this thesis. The principal researcher proposed two variants of the basic building blocks of the framework (i.e. the meta-model and the set of criteria). In the second variant, the meta-model and the set of criteria are designed to capture in a specific manner the relevant functional requirements of a software application by using the elements of the use-case model (i.e. UML use-cases and scenarios).
  Future work includes extending the domain of the framework, and in particular the design of the meta-model and set of criteria, to capture the relevant functional requirements by using elements from other UML models (e.g. component models) that capture and represent the functional requirements specifications at later phases of the development life cycle of software projects: this will improve the verification of the design of the scaling factors framework into a more rigorous one.

Further future work may also include the alignment of the design of the scaling factors framework with other software requirements modeling languages – other than the Unified Modeling Language (UML) – that are used in specific domains like systems engineering domains, and therefore this will improve the awareness, and the use of the scaling factors issue in other domains than domain of software development.

- **design of Unified Modeling Language (UML) profile:** a profile in the Unified Modeling Language provides a mechanism to extend and customize the UML models for particular domain and platforms, like system engineering applications and real-time and embedded applications, which allows to refine the standard semantics of the UML in an additive and non-contradictory manner (Si Sahir, 2002). In the literature, there are a few proposals to design a UML profile for the software measurement community to customize the Unified Modeling Languages for the needs of the personnel involved in the early approximation of software functional size. However, none of these proposals is fully verified to ensure its applicability with variety of case studies or its usability by different groups of practitioners in the software engineering industry.

  Furthermore, the proposals presented in the literature did not take into considerations the 'quality' issues that the functional requirements specifications may have, especially at early phases of the life cycle of software development projects like requirements incompleteness and the issue of identifying the correct level of granularity of such incomplete functional requirements.

  This thesis presents the design of the one of the primary building blocks of a UML profile (i.e. the two variants of the meta-models). Future work may include the design of the constraints which the meta-model (possibly) could not represent though its mechanisms, the design tag definitions and the design of the stereotypes.

BIBLIOGRAPHY

Abrahao, Silvia, et Emilio Insfran. 2008. « A Metamodeling Approach to Estimate Software Size from Requirements Specifications ». In *Proceedings of the 2008 34th Euromicro Conference Software Engineering and Advanced Applications*. (Parma, Italy), p. 465-475. IEEE Computer Society.

Abran, A., J.W. Moore, P. Bourque et R. Dupuis. 2004. *SWEBOK: Guide to the Software Engineering Body of Knowledge 2004 Version*. Los Alamitos, California: IEEE Computer Society.

Abran, Alain, Jean-Marc Desharnais et Fatima Aziz. 2005. « Measurement Convertibility - From Function Points to COSMIC-FFP ». In *15th International Workshop on Software Measurement*. (Montreal, Canada, 12-14 September), p. 227-240.

Abu-Talib, Manar, Adel Khelifi, Alain Abran et Olga Ormandjieva. 2008. « Assessment of Real-Time Software Specifications Quality Using COSMIC-FFP ». In *Proceedings of Software Process and Product Measurement*. (Munich, Germany), sous la dir. de Cuadrado-Gallego, Juan J., Ren Braungarten, Reiner R. Dumke et Alain Abran, p. 183-194. Springer-Verlag.

Abualkishik, Abedallah, Mohd Selamat, Abdul Azim Abd Ghani, Rodziah Atan, Jean-Marc Desharnais et Adel Khelifi. 2012. « A Convertibility Study on the Conversion between FPA and COSMIC for Real Time Systems ». In *2012 Joint Coference of the 22nd International Workshop on Software Measurement and the 7th International Conference on Software Process and Product Measurement*. (Assisi, Italy), p. 144-149.

A. Albrech. 1979. *Measuring application development productivity*. In Proc. Joint SHARE/ GUIDE/ IBM Applications Development Symposium. Monterey, California. pp. 83-92.

Anda, Bente, Dag I. K. Sjøberg et Magne Jørgensen. 2001. « Quality and Understandability of Use Case Models ». In *Proceedings of the 15th European Conference on Object-Oriented Programming*. (18-22 June), p. 402-428.

Arlow, Jim, et Ila Neustadt. 2005. *UML 2 and the Unified Process*. USA: Addison-Wesley, Pearson Education.

Azzouz, Saadi, et Alain Abran. 2004. « A proposed measurement role in the Rational Unified Process (RUP) and its implementation with ISO19761 – COSMIC ». In *Proceedings of Software Measurement European Forum - SMEF 2004*. (Rome, Italy, 28-30 January).

Bellur, U., et V. Vallieswaran. 2006. « On OO Design Consistency In Iterative Developement ». In *Proceedings of the 3rd International Conference on Information Technology: New Generations*. (Washington, USA), p. 46-51. IEEE Computer Society.

Bévo, V., G. Lévesque et A. Abran. 1999. « Application de la méthode FFP à partir d'une spécification selon la notation UML: compte rendu des premiers essais d'application et questions ». In *9th International Workshop Software Measurement*. (Lac Supérieur, Canada), p. 230-242.

Bittner, Kurt, et Ian Spence. 2003. *Use Case Modeling*. Addison-Wesley, Pearson Education.

Bjarnason, E., K. Wnuk et B. Regnell. 2011. « Requirements are slipping through the gaps - a case study on causes & effects of communication gaps in large-scale software development ». In *Proceedings of the 2011 IEEE 19th International Requirements Engineering Conference*. (Trento, Italy), p. 37-46. IEEE Computer Society

Boehm, Barry (767). 1981. *Software Engineering Economics*, 1st edition. Prentice Hall.

Boehm, Barry, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald J. Reifer et Bert Steece. 2000. *Software cost estimation with COCOMO II*, 1st edition. Prentice Hall, 544 p.

Bolloju, Narasimha, et Sherry X. Y. Sun. 2012. « Benefits of supplementing use case narratives with activity diagrams-An exploratory study ». *Journal of Systems and Software,* vol. 85, no 9, p. 2182-2191.

Booch, G., J. Rumbaugh et I. Jacobsen. 1999. *The Unified Modeling Language User Guide*. The Addison-Wesley Object Technology Series.

British-Computer-Society. 2000. *IT Projects: Sink or Swim*. UK: The British Computer Society, Computer Bulletin.

Brown, Bernice. 1968. *Delphi Process: A Methodology Used For the Elicitation of Opinions of Experts*. Santa Monica, California: The RAND Corporation.

Buhne, S., G. Halmans, K. Pohl, M. Weber, H. Kleinwechter et T. Wierczoch. 2004. « Defining requirements at different levels of abstraction ». In *12th IEEE International Requirements Engineering Conference*. (Kyoto, Japan, 6-11 September), p. 346- 347.

Chulani, Sunita. 1998. « Incorporating Bayesian Analysis to Improve the Accuracy of COCOMO II and Its Quality Model Extension ». University of Southern California.

Cockburn, A. 2000. *Writing effective use cases*. Addison-Wesley.

Condori-Fernández, Nelly, Silvia Abrahão et Oscar Pastor. 2007. « On the estimation of the functional size of software from requirements specifications ». *Journal of Computer Science and Technology,* vol. 22, n° 3, p. 358-370.

Condori-Fernandez, Nelly, Maya Daneva, Luigi Buglione et Olga Ormanjieva. 2010. « Experimental Study Using Functional Size Measurement in Building Estimation Models for Software Project Size ». In *2010 Eighth ACIS International Conference on Software Engineering Research, Management and Applications (SERA)*. (Montreal, Canada, 24-26 May), p. 276-282.

Conte, M., T. Iorio et L. Santillo. 2004. « E&Q: An Early and Quick Approach to Functional Size Measurement Methods ». In *Software Measurement European Forum – SMEF 2004*. (Rome, Italy).

Correa, Alexandre, et Claudia Werner. 2004. « Precise Specification and Validation of Transactional Business Software ». In *Proceedings of the 12th IEEE International Requirements Engineering Conference*. (Kyoto, Japan), p. 16-25. IEEE Computer Society.

COSMIC. 2007. *Advanced & Related Topics*. Common Software Measurement International Consortium <www.cosmicon.com/dl_goto.asp?id=60 >.

COSMIC. 2011. *Guideline for assuring the accuracy of measurements*. Common Software Measurement International Consortium. < www.cosmicon.com >.

Dedene, G., et M. Snoeck. 1994. « M.E.R.O.DE.: a model-driven entity-relationship object-oriented Development method ». *ACM SIGSOFT Software Engineering Notes,* vol. 19, n° 3.

Desharnais, J.-M., B. Kocaturk et A. Abran. 2011. « Using the COSMIC Method to Evaluate the Quality of the Documentation of Agile User Stories ». In *2011 Joint Conference of the 21st International Workshop on and 6th International Conference on Software Process and Product Measurement (IWSM-MENSURA)*. (Nara, Japan, 3-4 November), p. 269 - 272.

Desharnais, Jean-Marc, Alain Abran et Juan Cuadrado. 2006. « Convertibility of Function Points to COSMIC-FFP: Identification and Analysis of Functional Outliers ». In *2006 Joint Conferece of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement*. (Potsdam, Germany, 1-3 November), p. 190-205.

Diab, H., F. Koukane, M. Frappier et R. St-Denis. 2005. « µcROSE: Automated Measurement of COSMIC-FFP for Rational Rose Real Time ». *Information and Software Technology,* vol. 47, n° 3, p. 151-166.

Diev, Sergey. 2006. « Software estimation in the maintenance context ». *ACM SIGSOFT Software Engineering Notes,* vol. 31, n^{o} 2.

DPO. 2007. *Early & Quick Function Points Reference Manual – IFPUG version*. Rome, Italy: Data Processing Organisation.

Dzung, Dang Viet, et Atsushi Ohnishi. 2009. « Improvement of Quality of Software Requirements with Requirements Ontology ». In *Proceedings of the 2009 9th International Conference on Quality Software*. (Jeju, Korea), p. 284-289. IEEE Computer Society.

Eclipse. 2010. « Eclipse Modeling Project ». < http://www.eclipse.org/modeling/ >.

El-Attar, Mohamed, et James Miller. 2006. « AGADUC: Towards a More Precise Presentation of Functional Requirement in Use Case Model ». In *Proceedings of the 4th International Conference on Software Engineering Research, Management and Applications*. (Seattle, Washington, USA), sous la dir. de Society, IEEE Computer, p. 346-353.

Ellis, Timothy J., et Yair Levy. 2008. « Framework of Problem-Based Research: A Guide for Novice Researchers on the Development of a Research-Worthy Problem ». *International Journal of an Emerging Transdiscipline,* vol. 11, p. 17-33.

Engels, G., J. H. Hausmann, R. Heckel et S. Sauer. 2002. « Testing the Consistency of Dynamic UML Diagrams ». In *Proceedings of the 6th International Conference on Integrated Design and Process Technology (IDPT)*. (Pasadena, California, USA). Society for Design and Process Science.

España, S., A. González et O. Pastor. 2009. « Communication Analysis: a requirements elicitation approach for information systems ». In *21st international conference on advanced information systems*. (Amsterdam, Netherlands). Springer, Berlin.

España, Sergio, Nelly Condori-Fernandez, Arturo González et Óscar Pastor. 2010. « An empirical comparative evaluation of requirements engineering methods ». *Journal of the Brazilian Computer Society,* vol. 16, p. 3–19.

Fehlmann, Thomas, et Eberhard Kranich. 2012. « Quality of Estimations: How to Assess Reliability of Cost Predictions ». In *Joint Conference of the 22nd International Workshop on Software Measurement and the 7th International Conference on Software Process and Product Measurement*. (Assisi, Italy), p. 8-14.

GÉLOG. 2008. *A Proposed Measurement Etalon: C-Registration System*. Montréal, Canada: Software Engineering Research Laboratory, École de technologie supérieure.

GÉLOG. 2013. « COSMIC Entry Level Practitioners Certificate Holders ». < http://www.gelog.etsmtl.ca/cosmic-ffp/entry_level_holders.html >.

Gencel, Cigdem. 2008. « How to use COSMIC Functional Size in Effort Estimation Models? ». In *Proceedings of the International Conferences on Software Process and Product Measurement* (Munich, Germany), p. 196 - 207. Springer-Verlag.

Gencel, Cigdem, et Carl Bideau. 2012. « Exploring the convertibility betwen IFPUG and COSMIC Function Points: Preliminary Findings ». In *2012 Joint Coference of the 22nd International Workshop on Software Measurement and the 7th International Conference on Software Process and Product Measurement*. (Assisi, Italy, 17-19 October), p. 170-177.

Gencel, Cigdem, et Onur Demirors. 2008. « Functional size measurement revisited ». *ACM Transactions on Software Engineering & Methodology,* vol. 17, n° 3, p. 1-36.

Gene, Gene Golub, et F. Van Loan Charles (P. 53). 1996. *Matrix Computations*, 3rd edition. Baltimore: The Johns Hopkins University Press.

Goguen, J. A., et C. Linde. 1993. « Techniques for requirements elicitation ». In *Proceedings of IEEE International Symposium on Requirements Engineering*. (4-6 January), p. 152-164.

Gomaa, H., et D. Wijesekera. 2003. « Consistency in Multiple-View UML Models: a case study ». In *Proceedings of the 6th International Conference on the UML Consistency*. (San Francisco, USA).

Grau, G., et X. Franch. 2007a. *ReeF: Defining a Customizable Reengineering Framework*, 4495. Coll. « CAiSE 2007 and WES 2007 ». Heidelberg: Springer.

Grau, G., et X. Franch. 2007b. « Using the PRiM method to Evaluate Requirements Model with COSMIC-FFP ». In *IWSM-MENSURA 2007*. (Mallorca), p. 110-120.

Habela, P., E. Glowacki, T. Serafinski et K .Subieta. 2005. « Adapting Use Case Model for COSMIC-FFP Based Measurement ». In *15th International Workshop on Software Measurement - IWSM 2005*. (Montréal), p. 195-207.

Heeringen, H. van, E. van Gorp et T. Prins. 2009. « Functional size measurement - Accuracy versus costs - Is it really worth it? ». In *Software Measurement European Forum – SMEF 2009*. (Rome, Italy, 28 - 29 May).

Hussain, Ishrar, Leila Kosseim et Olga Ormandjieva. 2010. « Towards approximating COSMIC functional size from user requirements in agile development processes using text mining ». In *Proceedings of the Natural language processing and information systems, and 15th international conference on Applications of natural language to information systems*. (Cardiff, UK), p. 80-91. Springer-Verlag.

IFPUG20926. 2009. *Function Point Counting Practices Manual*. Princeton, NJ, USA: International Function Point Users Group.

INFSCI1024. 2011. *Banking information system in INFSCI1024: Information Systems Analysis undergraduate course*. USA: University of Pittsburgh.

IEEEComputer Society. 1998. *IEEE Std. 830 - IEEE Recommended Practice for Software Requirements Specifications*. New York: IEEE Computer Society.

ISO19761. 2011. *A Functional Size Measurement Method – ISO19761: COSMIC*. Geneva, Switzerland: International Organization for Standardization.

Jenner, M. S. 2001. « COSMIC-FFP and UML: Estimation of the Size of a System Specified in UML - Problems of Granularity ». In *4th European Conference on Software Measurement and ICT Control*. (Heidelberg), p. 173-184.

Jensen, Randall. 1983. « An Improved Macro level Software Development Resource Estimation Model ». In *5th ISPA Conference*. p. 88-92.

Jones, C. 1986. *A Short History of Function Points and Feature Points*. Burlington, Massachusetts Software Productivity Research.

Jones, C. 1997. *Applied Software Measurement*. McGraw Hill.

Kamata, Mayumi Itakura, et Tetsuo Tamai. 2007. « How Does Requirements Quality Relate to Project Success or Failure? ». In *15th IEEE International Requirements Engineering Conference*. (New Delhi, India), p. 69-78.

Karner, G. 1993. « Metrics for objectory ». Sweden, University of Linköping.

Kassab, M., O. Ormandjieva, M. Daneva et A. Abran. 2007. « Non-Functional Requirements: Size Measurement and Testing with COSMIC-FFP ». In *IWSM-Mensura 2007*. sous la dir. de Cuadrado-Gallego, J.J., R. Braungarten, R.R. Dumke et A. Abran Vol. 4895, p. 168-182.

Kassab, Mohamad, Maya Daneva et Olga Ormandjieva. 2008. « A Meta-model for the Assessment of Non-Functional Requirement Size ». In *Proceedings of the 2008 34th Euromicro Conference Software Engineering and Advanced Applications*. (Parma, Italy), p. 411-418. IEEE Computer Society

Kececi, N., et A. Abran. 2001. « An Integrated Measure For Functioal Requirements Correctness ». In *International Workshop on Software Measurement (IWSM 2001)*. (Montreal, Canada), p. 137-150.

Keefer, D., et S. Bodily. 1983. « Three-Point Approximations for Continuous Random Variables ». *Journal of Management Science,* vol. 29, p. 595-609.

Knauss, Eric, Christian El Boustani et Thomas Flohr. 2009. « Investigating the Impact of Software Requirements Specification Quality on Project Success ». In *Proceedings of the 10th International Conference on Product Focused Software Development and Process Improvement* (Oulu, Finland), sous la dir. de Bomarius, Frank, Markku Oivo, Päivi Jaring et Pekka Abrahamsson Vol. 32, p. 28-42. Springer Berlin Heidelberg.

Kruchten, Philippe. 2000. *The Rational Unified Process: an introduction*. USA: Addison Wesley.

Kujala, Sari, Marjo Kauppinen, Laura Lehtola et Tero Kojo. 2005. « The Role of User Involvement in Requirements Quality and Project Success ». In *Proceedings of the 13th IEEE International Conference on Requirements Engineering*. (Sorbonne, France), p. 75-84. IEEE Computer Society.

Kuzniarz, L., et M. Staron. 2003. « Inconsistencies in Students Designs ». In *The 2nd Workshop on Consistency Problems in UML-based Software Development*. (San Francisco, USA), p. 9-18.

Lange, Christian, et Michel Chaudron. 2004. « An empirical assessment of completeness in UML designs ». In *Proceedings of 8th International Conference on Empirical Assessment in Software Engineering*. (Edinburgh, Scotland), p. 111-121.

Lange, Christian F.J., et Michel R.V. Chaudron. 2007. « Defects in Industrial UML Models - A Multiple Case Study ». In *Workshop on Quality in Modeling (MODELS 2007)*. (Nashville, TN, USA), p. 50-64.

Lavazza, Luigi, et Vieri del Bianco. 2008. « Functional size measurement based on problem frames: a case study ». In *Proceedings of the 3rd international workshop on Applications and advances of problem frames*. (Leipzig, Germany), p. 44-47.

Leung, Felix, et Narasimha Bolloju. 2005. « Analyzing the Quality of Domain Models Developed by Novice Systems Analysts ». In *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences*. (Big Island, Hawaii) Vol. 7, p. 188-194. IEEE Computer Society.

Levesque, Ghislain, Valery Bevo et De Tran Cao. 2008. « Estimating software size with UML models ». In *Proceedings of the 2008 C^3S^2E conference*. (Montreal, Canada, May 12-13).

Lind, K., R. Heldal, T. Harutyunyan et T. Heimdahl. 2011. « CompSize: Automated Size Estimation of Embedded Software Components ». In *2011 Joint Conference of the 21st International Workshop on Software Measurement and 6th International Conference on Software Process and Product Measurement (IWSM-MENSURA)*. (Nara, Japan, 3-4 November), p. 86-95.

Lind, Kenneth, et Rogardt Heldal. 2009. « Estimation of Real-Time Software Code Size using COSMIC FSM ». In *Proceedings of the 2009 IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*. (Shenzhen, Guangdong, China), p. 244-248. IEEE Computer Society.

Lindland, OI., G. Sindre et A. Sølvberg. 1994. « Understanding quality in conceptual modeling ». *IEEE Journal of software engineering,* vol. 11, no 2, p. 42-49.

Marín, B., N. Condori-Fernández, O. Pastor et A. Abran. 2008. « Measuring the Functional Size of Conceptual Models in a MDA Environment ». In *20th International Conference on Advanced Information Systems Engineering Forum*. (Montpellier), p. 33-36.

Marín, Beatriz, Giovanni Giachetti, Oscar Pastor et Alain Abran. 2010. « A Quality Model for Conceptual Models of MDD Environments ». *Advances in Software Engineering,* vol. 2010, p. 17.

Meli, Roberto. 1997. « Early and Extended FP: A New Estimation Method for Software Projects ». In *IFPUG Fall Conference*. (Scottsdale, Arizona, USA), p. 15-19.

Meli, Roberto. 2000. « On the Applicability of COSMIC-FFP for Measuring Software throughout its Life Cycle ». In *11th European Software Control and Metric Conference* (Munich, Germany, April 18-20), p. 1-10.

Meli, Roberto, et Luca Santillo. 1999. « Function Point Estimation Methods: A Comparative Overview ». In *The European Software Measurement Conference (FESMA 99)*. (Antwerp, Belgium).

Mellor, Stephen, Marc Balcer et Ivar Jacoboson. 2002. *Executable UML: A Foundation for Model-Driven Architectures*. Boston, MA: Addison-Wesley Longman Publishing Corporation.

Moody, Daniel. 2005. « Theoritical and Practical Issues in Evaluating the Quality of Conceptual Models: Current State and Future Directions ». *Journal Data & Knowledge Engineering,* vol. 55, no 3, p. 243-276.

Nagano, S., et T. Ajisaka. 2003. « Functional metrics using COSMIC-FFP for object-oriented real-time systems ». In *13th International Workshop on Software Measurement*. (Montreal).

Nelson, E. 1967. *Management Handbook for the Estimation of Computer Programming Costs*. Santa Monica, California: System Development Corporation.

Ochodek, M., B. Alchimowicz, J. Jurkiewicz et J. Nawrocki. 2011. « Improving the reliability of transaction identification in use cases ». *Journal of Information and Software Technology,* vol. 53, n^{o} 8, p. 885-897.

OMG. 2006. *Object Constraint Language 2.0 Specification*. Needham, MA, USA: Object Management Group.

Ouwerkerk, J., et A. Abran. 2006. « An evaluation of the design of Use Case Points (UCP) ». In *Proceedings of the International Conference on Software Process and Product Measurement (MENSURA 2006)*. (Cádiz, Spain), sous la dir. de Abran, A., R. Dumke et M. Ruiz, p. 83-97. Publish Service of the University of Cádiz.

Park, R. 1988. *The Central Equations of the PRICE Software Cost Model*. 4th COCOMO Users' Group Meeting.

Pauli, Josh, et Dianxiang Xu. 2006. « Integrating Functional and Security Requirements with Use Case Decomposition ». In *Proceedings of the 11th IEEE International Conference on Engineering of Complex Computer Systems*. (Stanford, California, USA), sous la dir. de Society, IEEE Computer, p. 57-66.

Poels, G. 2003a. « Definition and Validation of a COSMIC-FFP Functional Size Measure for Object-Oriented Systems ». In *7th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*. (Darmstadt).

Poels, G. 2003b. « Functional Size Measurement of Multi-Layer Object-Oriented Conceptual Models ». In *9th International Object-Oriented Information Systems Conference*. (Geneva), p. 334-345.

Putnam, L., et W. Myers. 1992. *Measures of Excellence*. Yourdon Press Computing Series.

Roques, P. 2004. *UML in Practice*. Chichester: John Wiley & Sons.

RUP. 1999. « Wylie college example, course registration project ». < http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/ >.

Salger, Frank, Gregor Engels et Alexander Hofmann. 2009. « Inspection effectiveness for different quality attributes of software requirement specifications: an industrial case study ». In *Proceedings of the 7th ICSE conference on Software quality*. (Vancouver, British Columbia, Canada), p. 15-21. IEEE Computer Society.

Santander, V. F. A., et J. F. B. Castro. 2002. « Deriving use cases from organizational modeling ». In *Proceedings of IEEE Joint International Conference on Requirements Engineering*. (Essen, Germany), p. 32- 39.

Santillo, Luca. 2012. « Easy Function Points – 'Smart' Approximation Technique for the IFPUG and COSMIC Methods ». In *Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 7th International Conference on Software Process and Product Measurement*. (Assisi, Italy), p. 137-142.

Sellami, Asma, et Hanene Ben-Abdallah. 2009. « Functional Size of Use Case Diagrams: A Fine-Grain Measurement ». In *Proceedings of the 2009 4th International Conference on Software Engineering Advances*. (Porto, Portugal, 20-25 September), sous la dir. de Society, IEEE Computer, p. 282-288.

Si Sahir, S. 2002. *Guide to applying the UML*. Springer, 350 p.

Soubra, H., A. Abran, S. Stern et A. Ramdan-Cherif. 2011. « Design of a Functional Size Measurement Procedure for Real-Time Embedded Software Requirements Expressed using the Simulink Model ». In *In 2011 Joint Conference of the 21st International Workshop on Software Measurement and the 2012 6th International Conference on Software Process and Product Measurement*. (Nara, Japan), sous la dir. de Society, IEEE Computer, p. 76-85.

Soubra, Hassan, et Alain Abran. 2012. « A Refined Functional Size Measurement Procedure for Real-Time Embedded Software Requirements Expressed using the Simulink Model ». In *2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 7th International Conference on Software Process and Product Measurement*. (Assisi, Italy), p. 70-77.

Stiefel, Malcolm. 2008. *The Auction Package System*. Permission granted to copy and use this case study without royalty.

Tichenor, C. 1998. « The IRS Development and Application of the Internal Logical File Model To Estimate Function Point Counts ». In *IFPUG Fall Conference of Use, ESCOM-ENCRESS 98*. (Orlando, Florida).

Top, Ozden, Onur Demirors et Baris Ozkan. 2009. « Reliability of COSMIC Functional Size Measurement Results: A Multiple Case Study on Industry Cases ». In *Proceedings of the 2009 35th Euromicro Conference on Software Engineering and Advanced Applications*. (Patras, Greece), p. 327-334. IEEE Computer Society.

Trudel, Sylvie. 2012. « Using the COSMIC functional size measurement method (ISO 19761) as a software requirements improvement mechanism ». Montreal, Canada, École de Technologie Supérieure, Université du Québec.

Trudel, Sylvie, et Alain Abran. 2010. « Functional Requirement Improvements through Size Measurement: A Case Study with Inexperienced Measurers ». In *Proceedings of the 2010 8th ACIS International Conference on Software Engineering Research, Management and Applications*. (Montreal, Canada), p. 181-189. IEEE Computer Society.

Trudel, Sylvie, et Jean-Marc Lavoie. 2008. *uObserve Software Specification*. Montreal, Canada: Department of Software Engineering and Information Technology, École de Technologie Supérieure, Université du Québec.

Tsumaki, Toshihiko, et Tetsuo Tamai. 2006. « Framework for Matching Requirements Elicitation Techniques to Project ». *Software Process: Improvement and Practice,* vol. 11, n° 5, p. 505-519.

Vogelezang, Frank, et T. G. Prins. 2007. « Approximate size measurement with the COSMIC method: factors of influence ». In *Proceedings of the Sofware Metrics European Forum (SMEF 2007)*. (Rome, Italy, May 9-11).

Wang, Bo, Wei Zhang, Haiyan Zhao, Zhi Jin et Hong Mei. 2009. « A Use Case Based Approach to Feature Models' Construction ». In *17th IEEE International Requirements Engineering Conference*. (Atlanta, USA), p. 121-130.

Wu, Q., M. Ying, J. Wang, M. Xie, Z. Chen et J. Chen. 2009. « A discussion on three critical issues for assuring quality of the users' requirements specification ». In *16th International Conference on Industrial Engineering and Engineering Management*. (Beijing, China, 21-23 October), p. 690-693.

Xu, Jiandong, Tong Li, Zhongwen Xie et Tilei Gao. 2011. « Use Cases and Feedback in Functional Requirements Analysis ». In *2011 International Conference on Information Technology, Computer Engineering and Management Sciences (ICM)*. (Nanjing, China) Vol. 2, p. 54-57.