ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THESIS PRESENTED TO
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

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

BY
Mónica VILLAVICENCIO

DEVELOPMENT OF A FRAMEWORK FOR THE EDUCATION OF SOFTWARE MEASUREMENT IN SOFTWARE ENGINEERING UNDERGRADUATE PROGRAMS

MONTREAL, JUNE 17, 2014

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Alain Abran, Thesis Supervisor
Software Engineering and Information Technology Department at École de Technologie Supérieure

Mr. Mohamed Cheriet, President of the Board of Examiners
Automated Manufacturing Engineering Department at École de Technologie Supérieure

Mrs. Sylvie Rate, Member of the jury
Software Engineering and Information Technology Department at École de Technologie Supérieure

Mr. Hakim Lounis, External Evaluator
Imformatics Department at Université du Québec à Montréal

THIS THESIS WAS PRENSENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND PUBLIC

ON JUNE 10, 2014

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGMENT

There are several people who definitely deserve my sincere gratitude as they have been very supportive during my doctoral studies. It would have been impossible to develop the ideas presented in this dissertation without their support, comments, suggestions, and advice.

First, I would like to thank my thesis director, Dr. Alain Abran, as he has been an excellent reviewer and counselor, and a very supportive person; certainly, an example to follow. Many thanks to him for all the time he devoted to provide me guidance, as well as to review and comment all the documents produced through this thesis work.

I also thank Charles Symons for reviewing the case study and all the participants of my research studies for sharing with me valuable ideas, comments, knowledge, and expertise.

My thanks to my friends, colleagues and master students from ESPOL and ETS for all their help and support offered during my studies, especially Vanessa for developing the online version of the questionnaires used for my research studies.

Special thanks to my parents, brother and sister for all their kindness, support and love demonstrated at all times during my doctoral studies. Thanks a lot mom and dad for being with me every time I needed your help.

Finally, I express my immense gratitude to my love, Edgar, and my wonderful children: Andrés, Emily and María Belén. Edgar: I want to tell you that you deserve all my appreciation for your help, understanding, support and unconditional love. At last but not least, I am very grateful to God for having all of you in my life and for being my source of motivation and determination to reach this goal.

DÉVELOPPEMENT D'UN CADRE POUR L'ÉDUCATION DE LA MESURE DES LOGICIELS DANS LES PROGRAMMES DE GÉNIE LOGICIEL AU NIVEAU DU PREMIER CYCLE

Mónica VILLAVICENCIO

RÉSUMÉ

Les programmes de mesure des logiciels sont peu adoptés dans les organisations et il y a un manque d'attention à la mesure des logiciels dans l'enseignement supérieur. Ce travail de recherche vise à créer la base pour l'amélioration de l'éducation en mesure des logiciels dans les universités, en particulier dans les programmes de génie logiciel au niveau du premier cycle. Le but ultime de ce travail est de faciliter l'adoption de programmes de mesure dans les organisations produisant des logiciels.

Cette recherche aborde cette problématique en identifiant les sujets qui devraient être prioritaires pour les étudiants de premier cycle, et en élaborant un cadre éducatif sur la base de l'approche constructiviste et de la taxonomie de Bloom afin de fournir des lignes directrices pour les professeurs d'université.

Cette thèse inclue plusieurs activités de recherche, incluant un examen complet de la littérature, une enquête en ligne pour identifier les tendances actuelles dans l'enseignement de la mesure des logiciels, une étude Delphi pour identifier les priorités en matière d'éducation de mesure de logiciels pour les étudiants de premier cycle, et une évaluation du cadre pédagogique par des professeurs universitaires.

Les principaux résultats de ces études sont:

- Les experts dans le domaine ont identifiés cinq thèmes de mesure de logiciels essentiels (priorités) qui devraient être enseignés aux étudiants de premier cycle: les concepts de base de la mesure des logiciels, le processus de mesure, les techniques de mesure des logiciels, des mesures de gestion des logiciels, et des mesures pour la phase des exigences. Pour chacun de ces thèmes, les experts ont également identifié les niveaux d'apprentissage qui devraient être atteints par les élèves, selon la taxonomie de Bloom. De plus, les participants ont suggéré la nécessité d'inculquer aux élèves le développement de quatre compétences importantes au cours de leurs études universitaires, y compris: la pensée critique, la communication orale et écrite et le travail d'équipe. Ces compétences visent à compléter la connaissance et la pratique des élèves de la mesure du logiciel.
- La conception d'un cadre éducatif de la mesure du logiciel pour rencontrer ces exigences.

- Les professeurs d'université qui ont évalué le cadre proposé ont émis des avis favorables concernant son utilité pour l'enseignement de la mesure des logiciels et pour faciliter l'atteinte des résultats d'apprentissage par les étudiants de premier cycle.
- Un site Web conçu pour promouvoir l'éducation sur la mesure de logiciels http://software-measurement-education.espol.edu.ec/

Mots-clés: génie logiciel, la mesure des logiciels, l'enseignement supérieur, le constructivisme, la taxonomie de Bloom, cadre éducatif.

DEVELOPMENT OF A FRAMEWORK FOR THE EDUCATION OF SOFTWARE MEASUREMENT IN SOFTWARE ENGINEERING UNDERGRADUATE PROGRAMS

Mónica VILLAVICENCIO

ABSTRACT

Software measurement programs are hardly adopted in organizations and there is a lack of attention to software measurement in higher education. This research work aims at creating the basis for the enhancement of software measurement education in universities, specifically in software engineering programs at the undergraduate level. The ultimate goal of this work is to facilitate the adoption of software measurement programs in software related organizations.

This research project tackles this issue by identifying the software measurement topics that should be prioritized for undergraduate students and developing an educational framework on the basis of the constructivist approach and the Bloom`s taxonomy to provide guidelines to university teachers. By doing so, university teachers will be provided with tools and approaches to pursue the achievement of learning outcomes by students being introduced to software measurement tasks.

This research project required a number of investigations: a comprehensive literature review and a web survey to identify current practices in the teaching of software measurement; a Delphi study to identify priorities in software measurement education for undergraduate students; and an evaluation of the proposed educational framework by university teachers to determine the extent to which it can be adopted.

The key results are:

- Experts in the field agreed in identifying five essential software measurement topics (priorities) that should be taught to undergraduate students: basic concepts of software measurement; the measurement process; software measurement techniques; software management measures; and measures for the requirement phase. For each of these topics, the participating experts also identified the levels of learning expected to be reached by students, according to the Bloom's taxonomy. Moreover, they suggested the need for instilling in students the development of four important skills during their university studies, including: critical thinking; oral and written communication; and team work. These skills are aimed at complementing the students' knowledge and practice of software measurement.
- The design of an educational framework for the teaching of software measurement.
- University teachers evaluating the proposed framework gave favorable opinions regarding its usefulness for teaching software measurement and for facilitating the achievement of learning outcomes by undergraduate students.

- A website designed to promote the education on software measurement http://software-measurement-education.espol.edu.ec/

Keywords: software engineering, software measurement, higher education, constructivism, Bloom's taxonomy, educational framework

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABREVIATIONS

ACM	Association for Computing Machinery
CFP	COSMIC Function Point
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
DC	Degree of Consensus
DG	Data Group
DM	Data Movement
GQM	Goal question metric
GQIM	Goal question indicator metric
ISBSG	International Software Benchmarking Standards Group
PMBOK	Project Management Body Of Knowledge
PSM	Practical Software and Systems Measurement
PSP	Personal Software Process
SEEK	Software Engineering Education Knowledge
SE2004	Curriculum guidelines for software engineering undergraduate programs
SWEBOK	SoftWare Engineering Body Of Knowledge

INTRODUCTION

Measurement and quantitative analysis are fundamental in engineering. In the software engineering discipline, measurement is essential for software process improvement and certification purposes. Therefore, topics related to measurement are explicitly included in the existing software engineering curriculum guidelines and bodies of knowledge. Despite its importance, little has been done in academic settings to tackle the teaching of software measurement in undergraduate programs: in general, the literature related to software measurement in academia is focused on experimental studies to test techniques, tools and learning objects but with only slight attention in the achievement of learning outcomes by university students.

To tackle this problem, this thesis presents the design of an educational framework to facilitate the teaching and learning process of software measurement topics at the undergraduate level, which will address the key topics (priorities for undergrads) that students must learn in order to be able to support software measurement activities in the software organizations hiring them.

This thesis is organized into six chapters and 30 appendices.

Chapter 1 summarizes the literature review of the bodies of knowledge, curriculum guidelines, and previous academic initiatives related to software measurement. In addition, this chapter presents educational concepts and taxonomies that will be applied for designing the framework.

Chapter 2 presents the motivation for this research work, the objectives of the thesis, and a general view of the methodology designed to address them.

Chapter 3 presents the studies conducted prior the development of the framework, which provided information to build it according to the needs of university teachers and software measurement practitioners.

Chapter 4 describes the design of the proposed educational framework and provides one example of how to apply it for teaching software measurement to undergraduate students.

Chapter 5 presents the evaluation of the proposed framework and the studies conducted to perform such evaluation.

Chapter 6 presents the contributions of this research work, the foreseen improvements and future work.

Finally, the appendices includes eight publications: six for conferences and two for journals (one published and one submitted), the approvals of the ethics committee to conduct the studies with university teachers and practitioners, the questionnaires used for gathering information, extra examples of the applicability of the proposed educational framework; and suggestions for improvements to the current documents of the COSMIC-ISO 19761 software measurement method and suggestions for the related bodies of knowledge.

CHAPTER 1

SOFTWARE MEASUREMENT IN SOFTWARE ENGINEERING EDUCATION

This chapter summarizes the bodies of knowledge related to software measurement, the taxonomies and concepts associated to education, and previous work on the teaching of software measurement in higher education.

1.1 Bodies of Knowledge

This section presents three bodies of knowledge relevant for the education on software measurement: the Software Engineering Education Body of Knowledge (SEEK), the Software Engineering Body of Knowledge (SWEBOK), and the Software Measurement Body of Knowledge.

1.1.1 The curriculum guidelines SE2004

In 2004, the IEEE Computer Society and the Association for Computing Machinery (ACM) published guidelines for academic institutions and accreditation agencies to facilitate the design of undergraduate software engineering programs - SE2004 (IEEE ACM, 2004). These guidelines propose to educators the content of programs and the way of teaching them in different contexts (i.e. software engineering, computer science, and engineering). In addition, the guidelines suggest that a software engineer has to acquire technical and management skills to deal with software development processes from a holistic perspective as well as issues related to project management, product quality, standards and teamwork abilities. To instill these management and technical skills in undergraduate students, the Software Engineering Education Knowledge (SEEK) was introduced on the basis of the Software Engineering Body of Knowledge (SWEBOK 2004), the Project Management Body of Knowledge (PMBOK), the ACM report of Computing Curriculum, and two specific guidelines for undergraduate software engineering education (IEEE ACM, 2004).

Ten knowledge areas were selected to become part of the SEEK:

Computing Essentials

Mathematical & Engineering Fundamentals

Professional Practice

Software Modeling & Analysis

Software Design

Software Verification & Validation

Software Evolution

Software Process

Software Quality

Software Management

For each knowledge area of SEEK, the following information is provided in the curriculum guidelines SE2004:

1) Short description
2) Units and their topics
3) Number of suggested hours per unit
4) The level of knowledge per topic according to the taxonomy of Bloom. Three levels out of six were considered:
   - K: Knowledge
   - C: Comprehension
   - A: Application

The relevance of the topics

- E: Essential (part of the core)
- D: Desirable (if possible, it should be included as a core in specialized programs; or considered as elective)
- O: Optional (elective course)

In the SE2004, seven expected students' outcomes are proposed for an undergraduate curriculum in software engineering and are summarized into three, as follows:

- Acquiring knowledge (ie. models, techniques, software lifecycle, experience in different application domains) and skills (ie. teamwork, leadership, negotiation, good communication) to start working as a software engineer considering existing approaches as well as ethical, social, legal and economic concerns.
- Being capable to work under some constraints (cost, time, knowledge, etc) and exposed to changes in requirements.
- Becoming a self learner for continuous professional development.

Besides the outcomes, seven desirable characteristics of software engineers are stated. One of these is: "Engineers measure things, and when appropriate, work quantitatively; they calibrate and validate their measurements; and they use approximations based on experience and empirical data" (IEEE ACM, 2004). Due to these desirable characteristics, SEEK includes software measurement topics into their knowledge areas, which are shown in Table 1.1. The column L of this table shows the level of knowledge per topic according to the taxonomy of Bloom (K - knowledge, C - comprehension, A - application); and the column R represents the relevance of the topics (E - essential, D - desirable, O - optional). As it can be noticed from table 1.1, all the software measurement topics are considered as essential (E).

To the best of our knowledge, the curriculum guidelines for software engineering undergraduate programs (SE2004) have not been updated by the IEEE and ACM since 2004.

Table 1.1 SEEK Knowledge Areas, Units and topics related to measurement

SEEK Knowledge Areas	Knowledge Units	Topics	L	R
Mathematical and Engineering Fundamentals	Engineering foundations for software	Measurement and metrics	K	E
		Theory of measurement	C	E
Software Design	Design support tools and evaluation	Measures of design attributes (e.g. coupling, cohesion, etc.)	K	E
		Design metrics (e.g. architectural factors, interpretation, etc.)	A	E
Software Verification and Validation	V&V terminology and foundations	Metrics & Measurement (e.g. reliability, usability, etc.)	K	E
	Problem analysis and reporting	Analyzing failure reports	C	E
		Defect analysis	K	E
Software Process	Process concepts	Measurement and analysis of software processes.	C	E
		Quality analysis and control (e.g. defect prevention, quality metrics, root cause analysis, etc.)	C	E
	Process Implementation	Individual software process (model, definition, measurement, analysis, improvement)	C	E
		Team process (model, definition, organization, measurement, analysis, improvement)	C	E
Software Quality	Software quality processes	Software quality models and metrics	C	E
	Product assurance	Quality product metrics and measurement	C	E
Software Management	Project planning	Effort estimation	A	E
	Project control	Measurement and analysis of results	C	E

1.1.2 SWEBOK and the Software Measurement Body of Knowledge

SWEBOK (Software Engineering Body of Knowledge) is a guide that portrays the contents of the software engineering discipline, which provides a foundation for curriculum development (Abran, Bourque and Dupuis, 2004). The 2004 version of the guide includes 10 knowledge areas in which topics related to software measurement are dispersed all over them. Later, a proposal was put forward to have a full knowledge area (KA) to embrace software measurement; therefore, a Software Measurement Body of Knowledge was developed (Abran, April and Buglione, 2010). The document of this new body of knowledge, included in the encyclopedia of software engineering (Abran, April and Buglione, 2010), integrates and extends the measurement-related knowledge contained throughout SWEBOK 2004.

The Software Measurement Body of Knowledge distinguishes six major topics of software measurement: 1) Basic concepts; 2) Measurement process; 3) Measures by software life cycle (SLC) phase; 4) Techniques and tools; 5) Quantitative data; and 6) Measurement standards. Each of these topics is briefly explained: each has been divided into subtopics, and includes references.

At the time this thesis was written, a new version of the SWEBOK guide (SWEBOK v3) was under revision. This new version also includes software measurement topics spread throughout 15 knowledge areas - five new areas have been added since the 2004 version (Bourque, 2013).

Table 1.2 shows the relationship among the Software Engineering Education Knowledge (SEEK), the Software Measurement Body of Knowledge, and SWEBOK v3.

Table 1.2 Relationship of bodies of knowledge with respect to software measurement topics.

Software Engineering Education Knowledge (SEEK) and SE2004		Software Measurement Body of Knowledge		SWEBOK v3	
Knowledge Areas	**Topics**	**Major Topics**	**Sub Topics**	**Knowledge Areas**	**Topics**
Mathematical and Engineering Fundamentals	Measurement and metrics	Basic concepts	Foundations	Engineering foundations	Measurement
	Theory of measurement		Definition and concepts		
Software Design	Measures of design attributes (e.g. coupling, cohesion,etc.)	Measures by SLC	Software design	Software design	Software design quality analysis and evaluation
	Design metrics (e.g. architectural factors, interpretation, etc.)	N/F	N/F	N/F	N/F
Software Verification and Validation	Metrics & Measurement (e.g. reliability, usability, etc.)	Measures by SLC	Software testing	Software testing	Test related measures
	Analyzing failure reports				
	Defect analysis				
Software Process	Quality analysis and control (e.g. defect prevention, quality metrics, root cause analysis, etc.)	Techniques and Tools	Measurement techniques	Software engineering process	Software measurement
	Individual software process (model, definition, measurement,analysis, improvement)		Measurement techniques		Software process definition
	Team process (model, definition, organization, measurement,analysis, improvement)	N/F	N/F		Software process definition
Software Quality	Software quality models and metrics	Measures by SLC	Software quality	Software quality	Practical considerations - Software Quality Measurement
	Quality product metrics and measurement				
Software Management	Effort estimation	Measures by SLC	Software construction	Software engineering management	Software project planning
			Software testing		
		Techniques and tools	Measurement tools		
	Measurement and analysis of results	Measurement Process	Perform, Evaluate the measurement process		Software engineering measurement

N/F: Not Found

1.2 Educational taxonomies and constructivism

1.2.1 The Bloom's and SOLO Taxonomies

Since the appearance of the constructivist philosophy of education, a number of taxonomies have been proposed. Reigeluth and Carr-Chellman (2009) compared five of those taxonomies (1. Bloom, 2.Gagné, 3.Ausubel, 4.Merril, and 5.Reigeluth) and found that all of them share commonalities in learning outcomes. For example: the level of knowledge in the original taxonomy of Bloom is referred by Gagné as verbal information; by Ausubel as rote learning, by Merril as remember verbatim; and by Reigeluth as memorize information (Reigeluth and Carr-Chellman, 2009). Although several taxonomies have been created in the educational field, the most generally referred in educational research for higher education is the Bloom's taxonomy. Indeed, this taxonomy was used by ACM and IEEE to develop the curriculum guidelines for the software engineering and computer science university programs (IEEE ACM, 2004; Integrated Software & Systems Engineering Curriculum (iSSEc) project, 2009a; The Joint Task Force on Computing Curricula Association for Computing Machinery IEEE-Computer Society, 2013).

This doctoral work mainly used the revised version of the Bloom's taxonomy (Anderson et al., 2001) in order to be aligned with the curriculum guidelines and to be able to compare the further application of our educational framework -explained in chapter 4 - with other studies in higher education. Notwithstanding, this work also took into consideration the SOLO (Structure of the Observed Learning Outcome) taxonomy (Biggs, 1995; Biggs and Tang, 2007) to complement the Bloom's approach, especially in highlighting the importance of the constructive alignment and assessing the learning outcomes. The characteristics of both taxonomies (Bloom and SOLO) are presented in Table 1.3.

The Bloom's taxonomy has two dimensions: the knowledge dimension and the cognitive process dimension. The first dimension (knowledge) corresponds to the type of knowledge that students can learn: facts, concepts, procedures; and how they learn (metacognition). The

second dimension (cognitive) corresponds to the levels of learning that students can reach. This dimension has six levels of learning or categories, as follows (Anderson et al., 2001):

1) Remember: Retrieve information from long term memory.
2) Understand: Construct meaning from what the learner has heard, seen or read in class (lectures) or in books or computers.
3) Apply: Use a procedure in a given situation.
4) Analyze: Break material into its constituent parts, and determine how the parts relate to one another and to an overall structure or purpose.
5) Evaluate: Make judgments based on criteria and standards.
6) Create: Put elements together or reorganize them to create something new and functional.

Each of these levels is subdivided into cognitive processes, giving a total of 19 sub processes. For example: the *remember* category has two possible cognitive processes: recognizing and recalling. A complete list of cognitive processes is available on the cover pages of the Bloom's taxonomy book (Anderson et al., 2001).

The SOLO taxonomy is categorized into types of knowledge and levels of understanding. The types of knowledge are: declarative (i.e. knowing about, knowing *what*); and functioning (i.e. knowing *how* and *when*; that is, use declarative knowledge to solve problems, design, etc.). According to the authors of this taxonomy, as students learn, quantitative and qualitative changes are observed. Quantitative changes refer to the amount of details that students (learners) are able to provide; and qualitative changes refer to the integration of those details into a structural pattern. This means that the understanding becomes more structured and articulated as it develops. Then, this taxonomy considers five levels of understanding (Biggs, 1995; Biggs and Tang, 2007):

1) Prestructural: No understanding (information provided by students make no sense)
2) Unistructural: Few details are provided (information is simple and obvious)
3) Multistructural: More details are provided (quantitative increase)
4) Relational: There is a restructure or integration of components (qualitative increase)

5) Extended abstract: A new dimension of the relational level is perceived (connections that go beyond of the subject).

Table 1.3 Characteristics of the Bloom and SOLO taxonomies

Characteristics	**Revision of Bloom's taxonomy**	**SOLO taxonomy (Structure of the Observed Learning Outcomes)**
Terminology used for learning objectives	Educational objectives	Intended learning outcomes (ILO)
Ways to formulate learning objectives	VERB - Intended cognitive process; NOUN - Knowledge that students are expected to acquire or construct.	VERB - level of understanding or performance intended; CONTENT - the topics that the verb is meant to address; CONTEXT - the content discipline in which the verb is to be deployed.
Verbs for formulating learning objectives	There is a set of suggested verbs for every level of cognition.	A general list of verbs is provided without categorization.
Example of an objective/outcome	"Create a commercial about a common food product that reflects understandings of how commercials are designed to influence potencial clients"	"Design and develop particular constructs and models to support various levels of international business activities using different tools such as Microsoft Front Page, Microsoft access and Excel"
Constructivism alignment	Yes	Yes
Focus	Planning curriculum, instruction and assessment.	Qualitative assessment: study of the learning outcomes based on the details provided by students and their integration. Teaching and assessment approaches to achieve deep learning.
Levels of learning/understanding	Six levels: Remember, Understand, Apply, Analyze, Evaluate, and Create	Five levels: Prestructural; Unistructural; Multistructural; Relational; and Extended Abstract
Type of knowledge	Four types: Factual, Conceptual, Procedural, and Metacognitive	Two types: Declarative and Functioning
Teaching/learning activities	Examples are included but there is not a dedicated section for this subject.	The activities are classified according to the type of knowledge that students have to reach.
Assessment	General explanation of summative and formative assessment with some examples.	Explains and provides examples of summative and formative assessment related to the types of knowledge.

1.2.2 Constructive Alignment and Experiential Learning

The taxonomies presented above are based on the constructivist philosophy which has its origin in the studies of Jean Piaget, John Dewey, Lev Vigotsky, and Jerome Bruner (Pelech and Pieper, 2010). According to constructivism, the knowledge is not discovered; it is created by relating it with previous knowledge through personal experiences and social interaction. Moreover, the gain of knowledge is stimulated when learners are confronted with practical experiences and contextual problems (Anderson et al., 2001; Beard and Wilson, 2006; Biggs and Tang, 2007; Brooks and Brooks, 2001; Izquierdo, 2008; Lindsey and Berger, 2009; Pelech and Pieper, 2010).

The term constructive alignment refers to the tight relationship among the Intended Leaning Outcomes (ILO), the Teaching and Learning Activities (TLA), and the Assessment Tasks (AT). This means that learners can construct their own knowledge when the Teaching and Learning Activities (TLA) promote the Intended Learning Outcomes (ILO), and when the Assessment Tasks (AT) are used to verify the ILOs level of achievement (Biggs and Tang, 2007). Constructive alignment aims at helping the achievement of improved students learning outcomes through teaching environments that support the engagement of students with teaching activities and assessment tasks (Larkin and Richardson, 2012).

The application of constructive alignment seeks to achieve meaningful learning (deep, functioning) instead of surface learning (rote, declarative) in students. Meaningful learning is achieved when students can use relevant knowledge to solve new problems and to understand new concepts. Deep learning also entails the analysis of new ideas by taking into account existing knowledge (concepts, principles) (Anderson et al., 2001; Biggs and Tang, 2007; McAuliffe and Eriksen, 2011). Contrary to deep learning, surface learning implies having relevant knowledge without being able to use it to solve problems. According to McAuliffe and Eriksen, deep learning is "specially needed in disciplines in which the knowledge base and the methods for effective practice are less defined" (McAuliffe and Eriksen, 2011, p. 15).

This is the case of software measurement, which is still immature compared to other engineering disciplines.

The way in which learning is achieved relies on the instructional (teaching and learning activities) and assessment approaches used with students. In the case of surface learning, traditional ways of instructions (lectures, tutorials) and assessment (written exams) are suitable for achieving it. However, meaningful learning demands more involvement of students, which can be performed through games, simulations, role playing, real problems solving, field work, etc. Similarly, assessment tasks have an impact on learners. Reaching deep learning demands to put students' knowledge to work with real-life professional problems. This can be performed through projects (individual or in group), oral presentations, poster designs and presentations, reflective journals, development of case studies, and capstone projects (projects performed at the end of a university program).

As it can be noticed from the precedent paragraph, deep learning is associated with activities and tasks that may engage students in being active agents of their own learning. Learning by experience has its origins in the work of John Dewey, and has become popular with the experiential learning cycle of David Kolb (Beard and Wilson, 2006; McAuliffe and Eriksen, 2011). The cycle has four phases:

1) Concrete experience: Refers to an experience that stimulates the senses and alerts the learners. Concrete experiences are useful to introduce new concepts.
2) Reflective Observation: This is produced when the learner is able to observe and reflect on concrete experiences from many perspectives: that is, making sense from different angles. Teachers can promote reflection through class discussion or asking students to write a reflection paper.
3) Abstract conceptualization: This involves the creation or modification of guidelines, strategies, ideas or abstract concepts for taking action across situations. In this regard, teachers can explain concepts (abstract notions) and then provide examples, or provide examples first, and explain the concepts after.

4) Active experimentation: This implies the application or testing of concepts that have been learned through experience and reflection in further real-world experiences.

Summarizing, experiential learning takes place: when learners make connections between what they are learning and personal experiences; when learners connect what they are learning with what they know (previously learned); and when learners can apply what they have been taught to real-world problems (Beard and Wilson, 2006; Lindsey and Berger, 2009; McAuliffe and Eriksen, 2011). Active learning may demand from teachers the search of activities appealing to students followed by formative assessment, which together may have a long-term impact on students (learners).

Formative assessment is characterized by the provision of feedback to students during their learning process. That is, telling students how well they are doing and what needs to be improved in a task (project, exercise, presentation, etc) or in the process followed to perform that task, or clarifying concepts and procedures that were not completely understood (Anderson et al., 2001; Biggs, 1995; Biggs and Tang, 2007; Hattie and Timperley, 2007).

Formative assessment is very important to reach higher order levels of learning in students; however, the educational system is mostly oriented to perform summative assessment (grading students). Summative assessment often grades students at the end of a course (or fixed period) in an artificial condition - e.g. time pressure, written exam out of context (Biggs and Tang, 2007). This type of assessment is necessary for accreditation and policies making purposes; notwithstanding, it is advisable to avoid performing the assessment at the end of a course. If it is performed at the end, it will not be possible to provide feedback and help students to learn (Suskie, 2009).

1.2.3 Constructivist teaching

According to Pritchard and Woollard, constructivist teaching is "associated with learning that involves critical thinking, motivation, learner independence, feedback, dialogue, language,

explanation, questioning, learning through teaching, contextualization, experiments and real-world problem solving". To do constructivist teaching, these authors proposed the following seven principles (Pritchard and Woollard, 2010, p. 48):

1) "tell the learner why they are learning;
2) provide opportunities to make the learner feel in control;
3) provide opportunities for active engagement;
4) plan to use the learner's previous experiences;
5) plan to structure the learning experience based upon understanding of the curriculum;
6) be sensitive to emotional aspects of learning experiences;
7) contextualize the activities with real-life examples".

In addition, the authors suggest four forms of dialogue to be used in a constructivist classroom:

1) Assertions (e.g. definition of terms; introductions; teacher planning including learning outcomes and phases within the learning session, defining the scope; warm-up activities)
2) Contradictions (e.g. brainstorming; teacher contributions; debate)
3) Continuations (e.g. teacher-planned phases, action planning, roles within the groups)
4) Confirmation (e.g. group presentations; assignment writing; tests; teacher assessment questioning; summary; plenary).

Although, the literature offers advice for teaching in a constructive way, teaching may be seen as an art. According to Lupton, teaching should be personalized to respond to the students' needs. That is, the consideration of time constraints, learning environments, and teaching styles should guide teachers in the use of their creativity, originality, and innovation to teach in their own way. Therefore, Lupton suggests adding a fourth level to the existing three levels of teaching proposed by Biggs (2003): 1) who the student is; 2) what the teacher does; 3) what the student does; and 4) *who the teacher is* (Lupton, 2012).

Examples of the application of constructive alignment and associated concepts are presented in chapter 4.

1.2.4 Skills and their development

The use of experiential activities in constructivist teaching not only contributes to the acquisition of knowledge, but also to the development of skills (communication, teamwork, leadership, time management, critical thinking, etc.) and awareness (Beard and Wilson, 2006). Similar to learning knowledge, the learning of skills also requires experience and practice (Romiszowski, 2009). The process of acquiring skills is gradual, starting from narrow skills such as listening or questioning, and moving forward to broad skills such as teamwork, communication, critical thinking, etc. (Beard and Wilson, 2006).

Skill is defined by Romiszowski 2009 as "the capacity to perform a given type of task or activity with a given degree of effectiveness, efficiency, speed or other measure of quantity or quality". Romiszowski presents four types of skills: 1) cognitive or intellectual; 2) motor or psychomotor; 3) personal or reactive; and 4) interactive or interpersonal (Romiszowski, 2009).

- Cognitive or intellectual skills: require the use and management of the mind. Examples: problem solving, critical thinking, and decision making.
- Motor or psychomotor skills: refer to physical action, perceptual acuity and other skills that use and manage the body. Examples: playing a musical instrument, dance or athletic performance.
- Personal or reactive skills: handle of attitudes, feelings, and habits. Examples: self-expression, self-discipline and self-control.
- Interactive or interpersonal skills: entail social habits and the management of relationships and interactions with others. Examples: Active listening, persuasion, collaboration.

Some skills are more complex than others. That is, productive skills (e.g. thinking about a strategy to solve a problem) are more complex than reproductive skills (e.g. performing a repetitive or recurring activity). The development of complex - cognitive - intellectual skills requires knowledge. Therefore, to help students developing these skills, educators must

assure a deep understanding (deep-meaningful learning) of essential topics (Garrison and Archer, 2000; Romiszowski, 2009). According to Garrison et Archer (2000), there are four essential activities - that used in combination - promote meaningful learning: listening, talking, reading and writing.

Listening and reading (individually or in group) are needed for information acquisition, while talking and writing (individually or in group) are necessary for knowledge construction. In the case of listening (to a lecture, to a classmate doing an explanation, etc), it contributes to create meaning; whereas reading contributes to reflect about the subject in study. On the other hand, talking allows students to recognize contradictions, inconsistencies, and limited perspectives; writing leads students to think reflectively when organizing their ideas to produce a report, journal or other document. It is important to mention that listening has a low impact in learning when it is used alone. Listening should be complemented with the other three activities (reading, talking and writing) to increase the learning opportunities in students (Garrison and Archer, 2000).

This thesis work aims to facilitate the acquisition of software measurement knowledge in students and to reach the expected levels of learning by using constructivist teaching. However, the development of skills is indirectly targeted through the use of the four essential activities mentioned above (listening, reading, talking and writing) along the proposed educational framework designed and presented in chapter 4.

1.3 Initiatives in software measurement education

There are few works reported in the literature that tackle the education of software measurement in higher education, that is, publications that explicitly mention the interest of achieving or improving learning in students. Those works are presented next.

1.3.1 The METKIT project

The first work oriented to improve the education in software measurement was METKIT, an initiative financed with European funds through the SPRIT project. METKIT started in 1989 with the aim of designing educational material for both industrial and academic environments (Bush and Russell, 1992). At the time the METKIT work was conducted (1989-1992), there was a severe lack of material, especially text books that teachers and industrial educators could use in their software engineering courses. To identify the needs for software measurement teaching material, the METKIT team collected information from academia and industry across Europe, the United States, and Japan (6 countries in total). The research team conducted a postal survey among people from academia and industry and interviewed people from universities and educators from industry. Among the respondents, 69% of them agreed that there was not enough teaching material for software measurement at that time. Most of the interviewees expressed their desire to have available material related to software measurement principles, a set of examples and case studies instead of having a detailed prescription of explicit measures. Respondents also emphasized the need for students to recognize the importance of software measurement and to acquire skills for monitoring projects, predicting development effort, controlling the quality of a product, and documenting experience.

The teaching material, designed to help students and professionals in collecting simple but defined measures, was based on the Goal Question Metric (GQM) approach and the elaboration theory of Reigeluth and Stein (1983). The material consisted of two packages: academic and industrial. Both packages were developed in a modular form for the following reasons:

- To give students (from university and industry) an overview of the measurement domain before developing insights into a specific area;
- To avoid duplication of work while preparing material for industry and academia;
- To easily produce courses to individual organizations with specific training needs; and
- To incorporate modules in the existing courses taught at universities.

The academic package had 4 modules and the industrial 18. The academic package (object of interest for this thesis) covers the following topics:

- Module 0: Brief introduction to software engineering measurement.
- Module 1: Expanded the overview of Module 0 including the principles of measurement, definition of goals, data collection and analysis, the use of measurement in the industry, standards and certification.
- Module 2: It contained specialized topics such as: measurement theory, experimental design, a case study, and a tool sample (incorporated examples to run in software measurement commercial tools).
- Module 3: It gave guidelines to measure internal and external attributes.

These academic modules were targeted to undergraduate and postgraduate students specializing in software engineering or software engineering measurement.

In 1993, it was reported that 45 universities and 28 companies had bought the METKIT teaching packages (Bush and Ashley, 1993); however, usage has not be confirmed, and there has been little reference to this initiative since then.

1.3.2 The X-MED game

Another work related to measurement in an academic context is the exploratory study conducted by (Gresse von Wangenheim, Thiry and Kochanski, 2009). This work was performed to test the effectiveness of the game X-MED as an educational tool for complementing the lectures of a Software Measurement Module. X-MED was designed to meet the educational needs of graduate students in computer science programs and was tested among an experimental group of 15 students of two Brazilian universities. The duration of the module was 8 hours 45 minutes, including: a pre-test, lectures, in-class exercises, the game X-MED, and a post-test. Only the students who were part of the experimental group played the game. The tool is based on GQM and includes some elements of Practical Software and Systems Measurement (PSM). X-MED presented a hypothetical scenario in

which students have to develop a measurement program. During the game, students are automatically guided by following the steps of GQM. Each step is associated to a task that students need to perform by choosing an appropriate answer. Even when the students selected a wrong answer, the game provided feedback and continued leading them to the right way. On average the game lasts two hours. The expected levels of learning for this module according to the Bloom's taxonomy were:

- Remembering: Can the students recall information?
- Understanding: Can students explain ideas?
- Applying: Can students use a procedure?

It is important to remark that the exploratory study described above is the only one initiative that was identified in software measurement education in which the learning objectives were explicitly indicated and considered according to the Bloom's taxonomy. It is also relevant that the learning effectiveness was investigated. The authors had difficulties in demonstrating a positive learning contribution to master level students by using the X-MED game. It seems that the duration of the module was a major constraint to reach the expected levels of learning and to include and cover topics in depth.

1.3.3 The Play' n' Learn approach

Play' n' Learn is a general approach proposed by Buglione 2009. According to the author, the application of customized versions of well known games, like Monopoly, can serve to fix and verify the knowledge acquired by learners. Buglione customized a number of games to be used in software management and measurement training, with a focus on Functional Size Measurement (FSM). According to the author, the main reason for choosing FSM is that it represents one of the main inputs for determining the time and effort required for project planning (Buglione, 2009).

The approach (Play' n' Learn) includes guidelines with examples to create or customize games. The examples presented by Buglione consider a number of customized games with different levels of complexity: novice, intermediate and expert. The complexity is related to levels of knowledge that trainees have to achieve. Hence, the novice level deals with learning rules, awareness and basic knowledge; the intermediate or junior level has to do with notions; and the expert or senior level with managing concepts. Examples of games are:

- Level 1: Project-o-poly, this game highlights the need for certification and proper use of measures;
- Level 2: Trivial Pursuit (IFPUG FPA), game used to verify the knowledge of IFPUG FPA through questions and answers;
- Level 3: Taboo (IFPUG FPA), in this game learners have to guess a word (terminology of the Counting Practice Manual) based on five related words that appear in a card.

1.3.4 Visual learning techniques for software measurement

A recent work related to learning software measurement is the experiment conducted by Cuadrado et al. 2011, in which a visual environment was used to evaluate the effectiveness of visual over traditional methods in the learning of functional size measurement (Cuadrado-Gallego, Borja Martin and Rodriguez Soria, 2011). The experiment was performed with 43 students in their last year of a computer science degree program. The students took an initial exam and a final exam. They were divided into an experimental group and a control group. The experimental group of 18 students attended four sessions of visual learning methods lasting 1 hour and 30 minutes each. The control group attended four sessions lasting 2 hours each, given by a professor who taught them the theory of the measurement method. During the last session, the students in both groups performed a size measurement exercise. The experimental group obtained an average of 20 correct answers, compared to 15 for the control group in the final exam.

A second work of Cuadrado et al. related to the education of software measurement presents an analysis of the degree of dispersion in measuring the functional size of a real-world software application with undergraduate students (Cuadrado-Gallego et al., 2012). To carry out the experiment, the researchers trained and evaluated students before choosing the ones who qualified as participants for such experiment. A total of 97 undergraduate students registered in a Systems Planning and Management course received a 10 hours of theoretical lectures spread in five sessions of two hours. Only 61 students met the criteria for participating in the experiment, which were: having at least 90% of attendance in the training sessions; and demonstrating proper knowledge in COSMIC-ISO 19761 sofware measurement method by having a grade higher than 7/10 in a written exam -- 1 hour of duration. The exam had 50 multiple choice questions with four possible answers each. During the experiment, students used their knowledge to perform the measurement task in a maximum of 15 hours.

All the related works summarized above - as reported in the literature - present gaps in terms of: establishing the priorities with respect to the topics that should be emphasized in university courses; and the missing connection between the topics taught and the learning objectives. This doctoral thesis aims at filling this gap through the identification of priorities (see chapter 3) and the achievement of the learning outcomes (see chapter 4).

CHAPTER 2

RESEARCH OBJECTIVES AND METHODOLOGY

2.1 Research motivation

Between 2003 and 2008, I contributed to a research project, financed by Belgian universities, which aimed to help small and medium Ecuadorian software-development companies to improve their processes and to become aware of the new trends in the software engineering field. Within this R&D project, an inventory of companies was built to determine their size, the target market, the process they used to develop software, their beliefs and desire of obtaining certifications (ISO, CMMI), and so on. In addition, two workshops were organized with participants from industry and academia to develop strategies for reaching the project research objectives.

This led to changes in the curriculum of the software engineering courses taught at ESPOL University (Ecuador) at the undergraduate and graduate level, introducing subjects such as SPI (Software process improvement) and measurement techniques. Moreover, over that period, annual Software Engineering Conferences were held in Ecuador with the sponsorship of the Belgian universities and the IEEE. In 2008, Dr. Pierre Bourque visited ESPOL -as one of the invited speakers and talked about software measurement, including the COSMIC method and the ISBSG repository.

Several months later, officials from the Ecuadorian government who had attended the conference indicated to ESPOL's faculty their concerns about the lack of software development companies in Ecuador measuring the size of their products. The latter was intended to be a government requisite in order to make a proper estimation of prices of software applications. This governmental concern along with the desire of software companies to obtain CMMi certifications were one important motivation for the teaching of software measurement at the undergraduate level.

2.2 Research goal

The long term goal of this research work is to facilitate the implementation of measurement programs in software development organizations. We believe that the first step to reach this goal is the enhancement of education in software measurement at the undergraduate level, the undergraduates being the majority of the workforce in the software industry

2.3 Research objectives

To address the research goal, the following research objectives have been selected:

1) To determine the state of the art of software measurement in higher education.
2) To identify the priorities of software measurement in higher education at the undergraduate level.
3) To develop a framework for education in software measurement at software engineering undergraduate programs based on:
   - the related bodies of knowledge;
   - the results of the surveys and the Delphi study (see chapter 3);
   - the Bloom's taxonomy regarding the levels of learning (revised version (Anderson et al., 2001)); and
   - the constructivist approach.

2.4 Originality of the proposed research

As in any engineering discipline, measurement should be playing an important role in the software field. The connection between measurement and software process improvement has been emphasized, not only in the existing or upcoming standards, models and bodies of knowledge (Abran, Bourque and Dupuis, 2004; Bourque et al., 2008; IEEE ACM, 2004; Integrated Software & Systems Engineering Curriculum (iSSEc) Project, 2009b; Trienekens et al., 2007; Weber and Layman, 2002), but also in several studies conducted within the industrial software sector (Gopal et al., 2002; Iversen and Ngwenyama, 2006; Rainer and Hall, 2003; Staples and Niazi, 2008). Moreover, the curriculum guidelines for undergraduate

and graduate software engineering programs include knowledge areas in which measurement topics are explicitly considered (IEEE ACM, 2004; Integrated Software & Systems Engineering Curriculum (iSSEc) Project, 2009b).

Despite the extensive research work in software measurement and process improvement, little has been done in academic settings to tackle the lack of guidelines to address the teaching of software measurement in undergraduate programs. The reported efforts in addressing the education of software measurement are mostly focused in using specific methods, techniques or learning objects for gathering data from experimental studies conducted with students. Indeed, the shortcoming in providing proper education of software measurement in universities has been evidenced in a number of publications (Gresse von Wangenheim, Thiry and Kochanski, 2009; Jones, 2008; Zuse, 1998). Therefore, there is a need to consolidate the existing software measurement knowledge so that university students can learn and reach the expected levels of learning.

The present doctoral research work contributes to fill this gap by researching, designing and providing an educational framework to facilitate the teaching and learning process of software measurement topics considered as priorities in higher education at the undergraduate level.

By following the proposed framework, students (learners) are expected to become familiar with key software measurement topics as well as to be aware and able to support software measurement programs as part of software process improvement initiatives within organizations hiring them.

2.5 Target audiences of this research

The audiences targeted in this research are the following:

1) University teachers or instructors:
   - who want to include software measurement topics in their courses.

- who want to improve their current courses in which software measurement topics are included.
- who are interested in knowing how to apply active learning approaches in their classrooms and promoting meaningful learning among students.

2) Researchers looking for educational proposals that can be applied in higher education.
3) Software engineering practitioners or university students who want to learn software measurement by themselves and are looking for related theory and examples.
4) Software measurement practitioners, consultants or organizations looking for new examples of software measurement or new ideas for training.
5) Bodies of knowledge searching new publications and research works in their field of knowledge.

2.6 Overview of the research methodology

The methodology proposed to achieve the objectives of this doctoral thesis is divided into four phases:

- Phase 1: Literature review and design of surveys related to software measurement educational issues.
- Phase 2: Identification of priorities of software measurement for undergraduates.
- Phase 3: Design of an educational framework.
- Phase 4: Evaluation of the educational framework.

Figure 2.1 presents an overview of the research methodology, in which the inputs, phases, outputs and outcomes are visible. The corresponding chapters (sections, sub-sections) and appendices containing the details about the methodology appear in parenthesis.

The phase 1 includes the identification of software measurement topics taught at universities as well as the level of learning reached by students. The findings on this phase are based on

the reported studies published since the year 2000 (section 3.1) and on a new web survey designed and conducted among university teachers worldwide (section 3.2.1).

The phase 2 includes three activities to identify priorities in the education of software measurement: a web survey with practitioners (section 3.2.2), interviews with experienced software measurement teachers (section 3.4); and a Delphi study to reach consensus about the priorities (section 3.3).

The phase 3 includes: the definition of the educational framework; the design of its structure; and the filling-up of the framework. This latter consists of examples to show the applicability of the framework (chapter 4 and Appendix XV). The structure of the framework presents the connection of the required components to reach the expected levels of learning associated to the topics considered in the framework.

The phase 4 presents the evaluation of the framework performed by university students and teachers. The former were used to identify flaws in the understandability of examples and tasks assigned to students (sections 5.3.1 and 5.4.1); whereas the latter were important to determine the level of adoption of the proposed framework among teachers. The evaluation with teachers was done through a model adapted from (Gopal et al., 2002) -sections 5.2, 5.3.2 and 5.4.2).

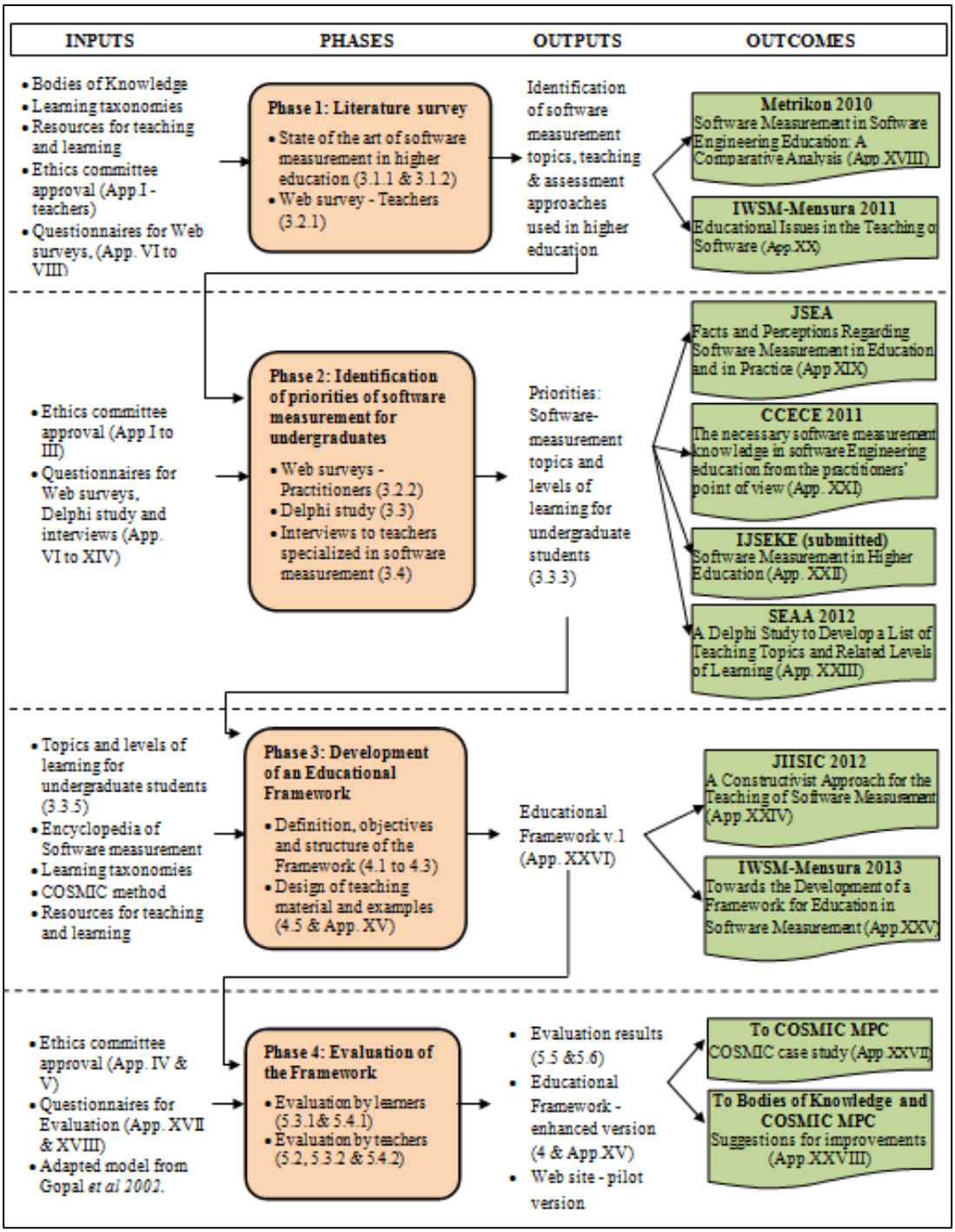

Figure 2.1 Overview of the research methodology

CHAPTER 3

STATE OF THE ART OF SOFTWARE MEASUREMENT IN HIGHER EDUCATION AND IDENTIFICATION OF PRIORITIES

This chapter presents the results obtained in the phases 1 and 2 of this research work: the literature survey and the research step taken for the identification of priorities of software measurement for undergraduates. The findings of these results will be used for the development of the educational framework explained in chapter 4.

3.1 Literature survey

3.1.1 The objective of the literature survey

The objective of this literature survey was to gain insights into how software measurement is taught at universities. To accomplish this objective, publications related to software measurement in academia were reviewed.

3.1.2 The selection of papers

The detailed information related to this literature survey (related work, methodology and results) is available in our first article "Software Measurement in Software Engineering Education: A Comparative Analysis" presented in Stuttgart, Germany in November 2010 (see Appendix XVIII) (Villavicencio and Abran, 2010).

This section briefly describes how we searched publications related to software measurement in an academic environment. This survey looked for studies conducted in universities where students – undergraduate or graduate - were performing measurement activities as part of their assignments in software measurement related courses (software engineering, software measurement, software quality, software project management, etc).

The databases used for this purpose were Compendex and Inspec. The searching criteria employed for executing the queries included a number of keywords, as follows:

Keywords:

- Software engineering;
- Measurement OR metrics;
- Education OR teaching;
- Experiment OR empirical;
- CMM OR PSP OR TSP OR GQM OR GQMI OR functional size OR COCOMO OR function points OR COSMIC OR McCabe OR estimation OR quality control.
- Undergraduate OR Graduate OR students
- Year of publication: from 2000 to 2010

In mid-2010, when this study was conducted, 18 articles met the search criteria. A complete description of the methodology used for this study can be found in section 3 of our article (see Appendix XVIII).

3.1.3 Results

The results published in (Villavicencio and Abran, 2010) can be summarized as follows:

- The majority of articles reporting experiments where students performed measurement activities (67%) were found in conference proceedings. From them, (28%) were published in the Software Engineering Education and Training Conferences – SEET.
- These experimental studies used mostly undergraduate (67%) rather than graduate students (33%).
- The experiments were mostly performed with students enrolled in computer sciences programs: Undergraduate level (67%) and graduate level (100%).
- For the experiments, students mostly worked with toy projects (undergraduate 75%, graduate 67%).

- The students participating in the studies were mostly enrolled in their last year of studies (undergraduate 67%, graduate 100%); and were taking a mandatory course related to software measurement (67% both undergraduates and graduates).
- In those experiments, students generally worked in teams (undergraduate 42%, graduate 67%).

  Only few articles mentioned the learning objectives related to software measurement (undergraduate 17%, graduate 33%).
- The topics most commonly covered according to the publications reviewed were:
  - Measurement techniques, mainly Goal Question Metric (GQM) and Personal Software Process (PSP) (undergraduate 50%, graduate 67%).
  - Measures by life cycle phase, especially effort, size and defects (undergraduate 42%, graduate 33%).
- The teaching approach most commonly used was lectures (50% both - undergraduates and graduates).
- Among the approaches used for assessing students' learning, written exams were explicitly referred (undergraduate 25%, graduate 17%).
- The level of learning expected to be accomplished by students was rarely specified (no more than 25% in both cases: undergraduates and graduates).

More details about the results are available in section 4 of Appendix XVIII. After the publication of our article, in 2011 and 2012, two new articles were identified (Cuadrado-Gallego et al., 2012; Cuadrado-Gallego, Borja Martin and Rodriguez Soria, 2011).

3.2 Web survey

From January to May of 2011, two web surveys were designed and conducted - one for teachers and the other for practitioners.

The survey administered to teachers was designed to build the state of the art of software measurement education from primary sources - the university teachers. On the other hand,

the survey administered to practitioners was intended to know issues related to software process improvement (SPI) initiatives and measurement programs in software organizations, and to get preliminary insights of the software measurement topics that should be the focus of university programs, according to practitioners.

The list of potential respondents was obtained from several sources, including: software measurement associations (i.e. GUFPI-ISMA); software measurement conferences organizers (i.e. UKSMA, IWSM-MENSURA); software engineering research group mailing lists (i.e. Competisoft), and digital libraries (i.e. engineering village and IEEE Xplore). A total of 159 respondents worldwide (107 teachers and 52 practitioners) answered the questionnaires.

Detailed information related to both surveys is included in Appendix XXII, which contains the article "Software Measurement in Higher Education" submitted to the International Journal of Software Engineering and Knowledge Engineering.

3.2.1 Web survey for teachers

3.2.1.1 Objectives of this survey

This survey aimed to identify:

- what software measurement topics are being taught in universities programs and their expected levels of learning.
- what practices are currently used at universities for teaching software measurement.

3.2.1.2 Methodology

The methodology for both surveys is explained in sections 3.1 to 3.5 of appendix XXII. The methodology mainly considers the design of the instrument (questionnaire) and the selection of the sample.

To design the instrument, we:

- created a list of software measurement topics based on the ACM and IEEE software engineering curriculum guidelines for undergraduate programs (IEEE ACM, 2004) and the Software Measurement Body of Knowledge (Abran, April and Buglione, 2010; Bourque et al., 2008).
- used the revised version of Bloom's taxonomy (Anderson et al., 2001) to propose a range of learning outcomes that are in relation with the six levels of learning of the taxonomy (remember, understand, apply, analyze, evaluate, and create).
- used concepts related to teaching and assessment approaches to write the questions.

To select the sample, we looked for university teachers who were teaching software engineering, software measurement, or any course in which software measurement topics are taught. The name and email of teachers were obtained from the sources mentioned in the section 3.2. From these, a total of 107 teachers - representing universities from 27 countries- answered the survey.

3.2.1.3 Results

The results from this sample are described in the Appendix XXII - section 4.1.

A brief summary of those results is presented next:

- The most referred courses in which software measurement topics are covered are: software engineering (48.6%), software quality (11.1%), software measurement (9.7%), and software project management (9%).
- The majority of the measurement-related courses (52.8%) is mandatory and taught to undergraduates.
- The measurement-related courses are generally offered during the third or fourth year of study for undergraduates (65.3%) and during the first year for graduates (77.6%).

- The software measurement topics most commonly covered at universities are: 1) basic concepts; 2) measures related to quality; 3) techniques and tools; and 4) the measurement process.
- In the Software Engineering courses, which represent the majority of courses where software measurement is taught, the main focus related to measurement relies on basic concepts and techniques and tools regardless the level in which they are taught (undergraduate or graduate).
- The first three levels of learning of the Bloom's taxonomy (remember, understand, apply) are expected to be achieved by all students, regardless of the educational level of the program (undergraduate or graduate).
- Graduate students are expected to achieve higher levels of learning (analyze, evaluate, and create) than undergraduates.
- Around 60% of teachers combine theory and exercises in class.
- Students work in groups to measure size, effort, and defects. They commonly measure toy software projects (64.5%).
- The instructional approaches most commonly used for teaching software measurement are: lectures (88.2%), case studies (64.6%) and collaborative activities (59.2%).
- The level of learning of students is commonly assessed through written exams (69.7%), term projects (67.1%) and presentations in class of assignments/projects (63.2%).

3.2.2 Web survey for practitioners

3.2.2.1 The objective of this survey

With this survey, we wanted to determine:

- the level of importance perceived by organizations on software measurement.
- how organizations appreciate software measurement knowledge acquired by graduating students when they become their employees.

- what specific software measurement topics should be emphasized in software engineering education from the practitioners point of view.

3.2.2.2 Methodology

The methodology applied to carry out this survey is similar to the one used for the web survey with university teachers (see section 3.2.1.2). However, for designing the questionnaire we took into consideration aspects related to software process improvement (SPI) initiatives and measurement programs in organizations. To create the questionnaire, some ideas were taken from articles related to surveys performed in the software industry such as (Bush and Russell, 1992; Chudnovsky, López and Melitsko, 2001; Salazar et al., 2004; Trienekens et al., 2007; Yazbek, 2010).

Regarding the sample, this was composed by practitioners working on SPI programs, as well as software measurement specialists from private or public organizations. A total of 52 practitioners from 18 countries answered the questionnaire.

3.2.2.3 Results

Partial results of this survey were presented and published in the 25th IEEE Canadian Conference on Electrical and Computer Engineering held in Montreal (see Appendix XXI). In addition, an extended version of this publication is included in the article "Software Measurement in Higher Education" submitted to the International Journal of Software Engineering and Knowledge Engineering (Appendix XXII - section 4.2).

A brief summary of the results is presented next:

- The majority of organizations represented in the sample had a Software Process Improvement (SPI) program (96%) at the time the survey was conducted.

- Among the certified organizations, 85% had ISO 9001 certification and 45% had CMMI. Certified organizations employ more people with Master's and Phd. degrees than non certified ones.
- From the set of organizations that had measurement programs, 89% of them used their own definitions of software measures, 22% used the ISO 20926 IFPUG 4.1 method, and 11% used the ISO 19761 COSMIC functional size method.
- The software measurement tools used by organizations are spreadsheets (41% - for registering their measures) and Microsoft Project for planning and tracking (26%).

According to the totality of this sample, there is an agreement in considering three software measurement topics as essential to be taught in university courses: basic concepts; the measurement process; and techniques and tools. For the rest of the topics, there were differences in opinions between respondents depending on the type of organization they work for (certified, not certified, with or without a measurement program). Respondents from certified organizations considered that measurement standards should be emphasized in university courses, while those from non certified organizations preferred software engineering management measures. In addition, certified organizations and those that had measurement programs gave greater importance to the topic *measures for the requirements phase* than non certified organizations and those without measurement programs.

3.3 The Delphi study to identify priorities

The Delphi method is used to reach consensus among experts regarding an issue that needs to be investigated or solved. For reaching consensus, several rounds are performed via a structured communication process (Amos and Pearse, 2008; Bourque et al., 2002; Gatchell, Linsenmeier and Harris, 2004; Howze and Dalrymple, 2004; Hung, Altschuld and Lee, 2008; Okoli and Pawlowski, 2004).

Delphi studies are generally used in educational research projects and are helpful for determining learning goals (Suskie, 2009).

Our Delphi study started in summer 2011 (preparation phase) and ended in fall 2012 (verification phase). The methodology and the results of the pilot test of this study are available in Appendix XIII, which corresponds to the article "Software Measurement in Software Engineering Education: A Delphi Study to Develop a List of Teaching Topics and Related Levels of Learning" presented in the 38th Euromicro Conference on Software Engineering and Advanced Applications - SEAA 2012.

3.3.1 The objective of the Delphi study

The objective of this study was to identify the software measurement topics that should be emphasized at the undergraduate level in software engineering programs, and the levels of learning that students should reach according to Bloom's taxonomy.

3.3.2 Methodology

As previously mentioned, the methodology is covered in Appendix XXIII (section III - Research Methodology). Figure 3.1 summarizes the steps followed to perform this Delphi study.

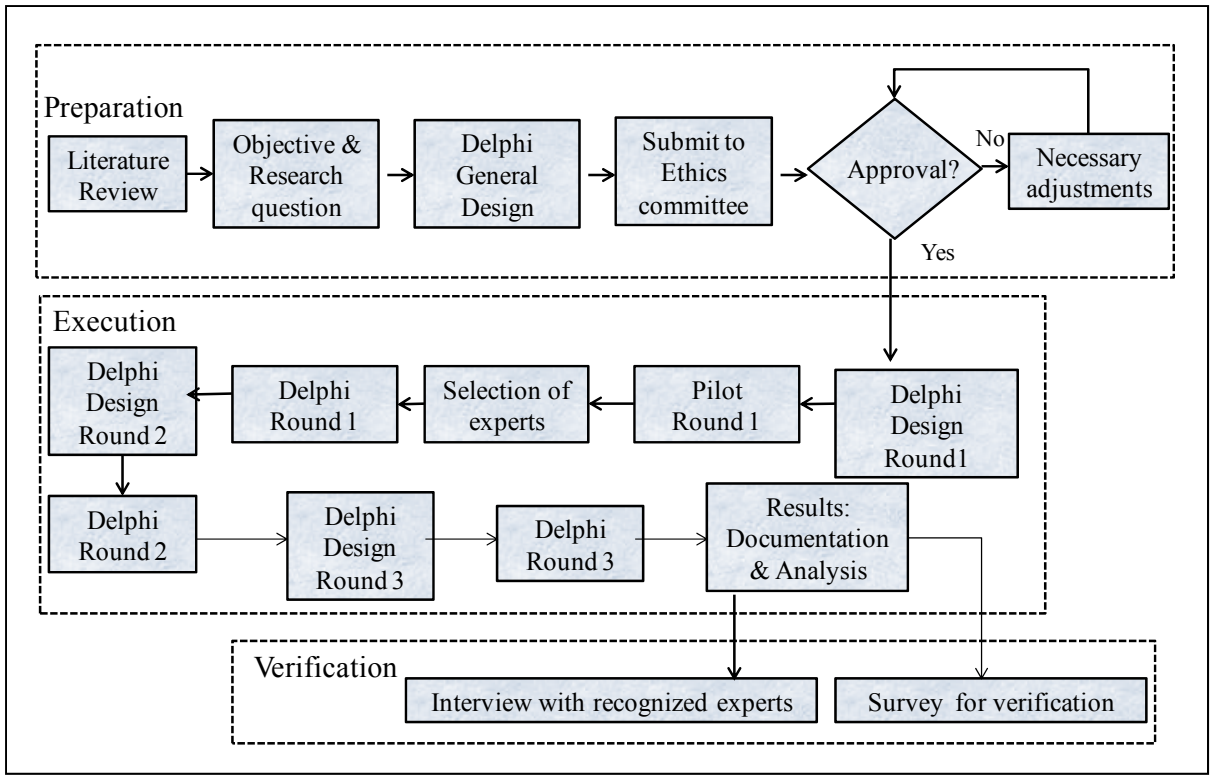

Figure 3.1 General view of the Delphi study - adapted from (Okoli and Pawlowski, 2004)

The Delphi study had two panels of software measurement experts: university teachers and practitioners. The profile of each type of participant is as follows:

Practitioners

- With five or more years of professional experience in software measurement by working on SPI programs, and/or software measurement programs as a team member or specialist.
- Members of a software measurement committee or a software measurement association (not mandatory, but preferable).
- With publications in software measurement related to professional and research experience in the field.
- With post secondary education.

University teachers

- With five or more years of teaching experience in specialized software measurement courses, or related software engineering courses in which software measurement topics are covered.
- With publications in software measurement related areas.
- Members of a software measurement committee or a software measurement association (not mandatory, but preferable).

A list of potential participants who met the expert profiles was developed by searching articles related to software measurement in digital libraries (e.g. IEEE Xplore, Engineering Village) and by looking for experienced practitioners in LinkedIn specialized software measurement groups.

Our Delphi study was performed in three rounds:

1) In the first round, the experts had to choose five software measurement topics that they considered as essential to be taught at the undergraduate level; they also selected the expected levels of learning per each of the topics; and the set of skills required for undergraduate students to complement their education in software measurement.

In the second round, the order of importance of the software measurement topics, skills and levels of learning was determined by the participants.

In the third round, participants had to manifest whether or not they agreed with the ranking of the priorities defined in round 2 or provide new rankings.

Table 3.1 shows the number of participants who were invited per panel and the number of participants per round.

Table 3.1 Number of participants of the Delphi study

Rounds	Teachers' panel	Practitioners' panel	Both panels
Participants invited	31	34	65
Round #1	17	18	35
Round #2	14	15	29
Round #3	14	16	30

After each round, a summary of the results was sent to the participants along with the invitation to participate in the next round.

After the three rounds, the results were verified by two means: 1) conducting a survey among people attending software engineering international conferences (practitioners and teachers); and 2) interviewing recognized experts in the software measurement field.

In the case of the participants of the survey for the verification step after the three Dephi rounds, they did not have to meet selection criteria to fill out the questionnaire: the participation in this verification process was completely voluntary. A total of 50 people answered the verification questionnaire (26 teachers and 24 practitioners).

Regarding the recognized experts, the profile defined was:

- A person with more than 10 years of experience in the field by leading software measurement communities, associations, or enterprises specialized in software measurement.
- With relevant publications in software measurement such as books and/or articles in renowned journals.

By searching software measurement books in Amazon.com, eleven recognized experts were identified. Four of them were interviewed in a software measurement conference held in Italy in October 2012.

3.3.3 Results

3.3.3.1 Results from rounds 1 to 3

This section is divided in three parts:

- Software measurement topics,
- Levels of learning, and
- Complementary skills that need to be considered in software measurement education.

Software measurement topics

During the first round, the participants of each panel were asked to select among a list of software measurement topics from Phase 1 the five most important to be taught to undergraduate students. Each topic included in the list had examples in order to avoid misunderstandings or confusion among participants.

With the data of each panel, we used the following criteria to select the five most important topics:

- More than 50% of the experts in both panels (university teachers and practitioners) chose the topic.
- More than 50% of one expert‘s panel chose the topic (university teachers or practitioners).
- The topic did not reach the 50% acceptance level but it was still rated among the 5 most important topics in both panels.

Both panels agreed on four topics. However, each panel selected a different fifth topic, given a total of six topics that met any of the above criteria, as follows:

1) Basic concepts in software measurement (both panels)
2) The measurement process (both panels)
3) Techniques and tools for software measurement (both panels)
4) Software management measures (both panels)
5) Measures for the requirements phase (practitioner's panels)
6) Measures for the design phase (teachers' panel)

It is worth mentioning that the topics were arranged in a random order in the questionnaires used for each round.

In rounds 2 and 3, the participants were asked to rank the topics. The criteria that we used to determine the ranking provided by the participants were:

- The mathematical mode (the ranking most commonly selected by participants for a specific topic)
- The number of votes per topic.

The degree of consensus (**DC**) of participants was defined as follows:

- Less than 10% votes (0-9.99%): Very weak degree of consensus (VW)
- Less than 30% votes (10-29.99%): Weak degree of consensus (W)
- Less than 50% votes (30-49.99%): Moderate degree of consensus (M)
- Less than 70% votes (50-69.99%): Strong degree of consensus (S)
- Less than 90% votes (70-89.99%): Very strong degree of consensus (VS)
- Less than or equal to 100% votes (90-100%): Extremely strong degree of consensus (ES)
- With an undetermined mode: No consensus

In round 2 of the teacher's panel, the order of the first four positions of the rankings was identified. Notwithstanding, positions 5 and 6 were not clear. A third round was required to confirm the four first positions and to identify positions 5 and 6.

In round 2 of the practitioners' panel, the positions 1 and 4 to 6 were identified but not the positions 2 and 3. The round 3 was needed to identify those positions with low level of agreement.

Table 3.2 presents the final rankings of the software measurement topics obtained in round 3. More information about these results, including the explanations provided by participants, is available in Appendix XXIX.

Table 3.2 Ranking of software measurement topics for undergraduates

TOPICS	TEACHERS			PRACTITIONERS		
	Ranking	% votes	DC	Ranking	% votes	DC
Basic concepts in software measurement	1	79%	VS	1	80%	VS
The measurement process	2	79%	VS	2	81%	VS
Techniques &Tools	3	71%	VS	3	69%	S
Software management measures	4	71%	VS	4	81%	VS
Measures for the requirements Phase	5	64%	S	5	81%	VS
Measures for the design phase	6	57%	S	6	75%	VS

Levels of learning

In the previous section on measurement topics, we explained how the participants chose the five most important topics. In this section on levels of learning, we explain how the levels of learning for each of those topics were determined.

In the first round, the participants had to select from a list of levels of learning, the ones that - according to them - should be reached by undergraduate students in the five most important software measurement topics.

A web application was developed in such a way that once the participants had selected the five most important topics, their corresponding levels of learning appeared in the screen to be selected. Every topic had between 4 to 6 levels of learning related with the levels of the Bloom's taxonomy. This means that every expected level of learning per topic had a corresponding level of the Bloom's taxonomy. The participants did not know the relationship of the levels of learning shown in the screen with the Bloom's taxonomy. In the analysis, we did use the relationship to match the levels of learning selected by participants with the Bloom's taxonomy.

Once the participants had selected the levels of learning per topic (the five most important), the following criteria were used for identifying the most preferred:

- More than 50% of the experts in both panels chose the level of learning.
- More than 50% of one expert‘s panel chose the level of learning.
- The level of learning did not reach more than 50% of acceptance, however it was considered as the most important in both panels.

For the second round, we asked again the participants to choose the levels of learning per topic. In this round we got a clear consensus about the preference of the participants (strong, very strong, extremely strong), so we did not include this question in round 3.

The criteria used to categorize the preferences of the participants regarding the levels of learning were:

- Less than 10% votes (0-9.99%): Very weak preference (VW)
- Less than 30% votes (10-29.99%): Weak preference (W)
- Less than 50% votes (30-49.99%): Moderate preference (M)
- Less than 70% votes (50-69.99%): Strong preference (S)

- Less than 90% votes (70-89.99%): Very strong preference (VS)
- Less than or equal to 100% votes (90-100%): Extremely strong preference (ES)

Table 3.3 shows the levels of learning per topic preferred by participants. As it can be noticed, the first levels are the most preferred. Extended information can be found in Appendix XXIX.

Complementary skills that need to be considered in software measurement education

To complete the information related to the education of software measurement, a section about skills was included in the Delphi study. In the first round, participants were asked to select -among a list - four skills needed to complement the education of software measurement at the undergraduate level.

The criteria used to select the skills that passed to round 2 were the following:

- More than 50% of the experts in both panels chose the skill.
- More than 50% of one expert‘s panel chose the skill.

The skills selected in round 1 were:

- Oral communication
- Written communication
- Team work
- Critical Thinking

Table 3.3 Preference of Levels of Learning per topic and panel

Levels of Learning per software measurement topic	TEACHERS		PRACTITIONERS	
	% Pref.	Degree Pref.	% Pref.	Degree Pref.
BASIC CONCEPTS IN SOFTWARE MEASUREMENT				
L1. Remembering software measurement terminology and concepts	100	ES	80	VS
L2. Giving examples of basic concepts, measurement methods and procedures	100	ES	80	VS
L3. Explaining them	79	VS	60	S
L4. Using terminology and concepts in a given exercise/project	57	S	60	S
THE MEASUREMENT PROCESS				
L1. Remembering the measurement process	86	VS	73	VS
L2. Explaining the measurement process	100	ES	73	VS
L3. Using the measurement process in a given project/situation	57	S	53	S
L4. Designing/modifying a measurement process for a specific situation	21	W	13	W
TECHNIQUES AND TOOLS FOR SOFTWARE MEASUREMENT				
L1. Remembering the existing software measurement techniques and tools	86	VS	80	VS
L2. Giving an example of them	93	ES	73	VS
L3. Explaining them	79	VS	60	S
L4. Following a technique in an exercise or in a given project	79	VS	53	S
L5. Using an appropriate technique/tool (according to what is needed in a given project/situation)	36	M	33	M
SOFTWARE MANAGEMENT MEASURES				
L1. Remembering concepts related to effort estimation, measures for project planning and control	93	ES	80	VS
L2. Explaining how to estimate the effort of a project	86	VS	67	S
L3. Measuring time and effort in a given project	57	S	67	S
L4. Using effort estimation models according to the situation	36	M	27	W
L5. Analyzing measurement results for project control	29	W	40	M
MEASURES FOR THE REQUIREMENTS PHASE				
L1. Remembering the most common functional size measurement methods	93	ES	80	VS
L2. Interpreting how the most common functional size measurement methods work	79	VS	67	S
L3. Obtaining the functional size of the software in an exercise or project by following a measurement method	64	S	73	VS
L4. Using an appropriate functional size measurement method (according to what is needed in a given project/situation)	43	M	40	M
MEASURES FOR THE DESIGN PHASE				
L1. Remembering the measures for the design phase	86	VS	80	VS
L2. Interpreting the meaning of the measures and how to obtain them	71	VS	67	S
L3. Obtaining the measures in an exercise or in a given small project	57	S	47	M
L4. Using the measures when it is appropriate in a real/simulated situation	36	M	20	W

In rounds 2 and 3, the participants were asked to rank the skills. The criteria that we used to determine the ranking were the same used for the topics.

In round 2 of the teacher's panel, only the positions 1 and 2 of the ranking were identified. So, we needed the third round to identify the remaining positions.

In round 2 of the practitioners' panel, the positions 3 and 4 were identified but not the first two of the ranking. The third round was also needed.

The table 3.4 presents the final rankings of the skills obtained in round 3. Appendix XXIX contains complementary information.

Table 3.4 Ranking of skills needed to complement education in software measurement

SKILLS	TEACHERS			PRACTITIONERS		
	Ranking	% votes	DC	Ranking	% votes	DC
Critical Thinking	1	57%	S	1	75%	VS
Oral communication	2	100%	ES	4	94%	ES
Written communication	3	86%	VS	3	53%	S
Teamwork	4	86%	VS	2	69%	S

3.3.3.2 Results from the verification phase

Survey among people attending software engineering international conferences

The main objective of the survey was to verify the results of the Delphi study. To accomplish this objective this survey was conducted among teachers and practitioners.

The survey was promoted among the attendees of the following conferences:

- 38th Euromicro Conference on Software Engineering and Advanced Applications. September 5-8, 2012. Cesme, Izmir, Turkey.
- The Joint Conference of the 22nd International Workshop on Software Measurement (IWSM) and the 7th International Conference on Software Process and Product Measurement (Mensura), October 17-19 2012, Assisi, Italy.
- The IX Ibero American Conference on Software Engineering and Knowledge Engineering, November 28-30, 2012, Lima, Peru.

Fifty people from 18 countries showed interested in our research work and voluntarily answered the questionnaire: 26 teachers and 24 practitioners.

For the survey, similar questionnaires were used with both types of participants. The questionnaires for teachers and practitioners had a general information section and three sections to verify the results of the Delphi study (software measurement topics, levels of learning; and skills to complement the education in software measurement) - See Appendix XIII. The questionnaire for teachers included an additional section to gather information about educational aspects related to constructivism and active learning. This additional section and the information collected during the interviews with teachers provided relevant inputs to develop the framework presented in chapter 4.

For the three sections used for the verification of the Delphi study, the respondents had to indicate their level of agreement with the results obtained by using a five points Likert scale from *strongly agree* to *strongly disagree*. The results presented here and the following pages correspond to the counts and percentages obtained for the alternative *strongly agree*.

As it can be noticed from table 3.5, the topics *techniques and tools* and *measures for the requirement phase* obtained less percentages of *strongly agree* among practitioners. However, in the case of *techniques and tools*, the majority of practitioners chose *somewhat agree* (45.8%); giving an agreement of 75% (strongly agree + somewhat agree) in the ranking of this topic. In the case of *measures for the requirement phase,* 33.3% of

practitioners chose *somewhat agree* (i.e. strongly agree + somewhat agree = 75%). The reasons given by practitioners to justify their selection are listed below:

- In real projects, the requirements phase has the strongest impact in project success. IT students should be more convinced to study this topic than techniques and tools.
- You first need to know what the specific "measures" are prior to selecting tools. Measures tell you "what" while tools show you "how".
- Measures for design (#6) are specific; they are suitable for master degree. There are several methods for the measure #5, explaining all of the function point counting methods would last too long. It would be suitable in master degree not for undergraduates.
- The position #5 (measures for the requirements phase) should be #1 because all the problems start with poor requirements.
- The position #5 should be #3 because it is better to know these concepts before introducing the #4 (software management measures).
- The measures in the requirements phase are more relevant than management measures.
- The ranking reflects a theoretical approach in software measurement. In my experience, it is better to give an objective first with "why we need measures" by following a pragmatic approach. So I would rather start with examples of measures and why we need them and how they can be used and then the final stage I would go for the theory and fundamentals of measurement. My ranking would be 4, 5, 6, 3, 1, 2.

Table 3.6 shows the results of the preference of levels of learning chosen by teachers and practitioners. For simplification purposes, this table only includes the levels of learning that reached 50% or higher percentages in the category *strongly agree*.

Table 3.5 Verification of the ranking of the software measurement topics

Ranking	Topics	Strongly agree			
		Teachers (N1=26)		Practitioners (N2=24)	
		#	%	#	%
1	Basic concepts	23	88.5	23	95.8
2	Measurement process	18	69.2	17	70.8
3	Techniques and tools	17	65.4	7	29.2
4	Software management measures	13	50.0	12	50.0
5	Measurement for the requirements phase	15	57.7	10	41.7
6	Measurement for the design phase	13	50.0	12	50.0

Disagreements are observed in some levels of learning of three topics: Techniques and tools; measures for the requirements phase; and measures for the design phase. Similar discrepancies were detected during the Delphi study (see table 3.3); especially with regard to the topic Measures for the design phase, which is generally preferred by teachers but not by practitioners. This seems to suggest that teachers prefer to teach measures related to programming tasks with which the students are more familiarized.

Regarding the skills needed to complement the education of software measurement for undergraduate students, the verification results show that - in general - the participants agree with the rankings (see table 3.7). However, the written communications skill did not reach a strongly agreement among practitioners. The justifications provided were:

- Almost every large problem in a real project comes from people who have communication and team work problems.
- I agree more or less, but I would put critical thinking less.
- Regarding writing skills, not all aspects are documented.
- Written and oral communications should have higher priority because I do not think critical thinking and team work can be easily improved. So, I would work on the "low-hanging fruit" first.

Table 3.6 Verification of the selection of levels of learning per topic

Levels of Learning per software measurement topic	Strongly agree			
	Teachers		Practitioners	
	#	%	#	%
BASIC CONCEPTS IN SOFTWARE MEASUREMENT				
Can remember software measurement terminology and concepts	19	73.1	17	77.3
Can give examples of basic concepts, measurement methods and procedures	22	84.6	16	72.7
Can explain the above	20	76.9	12	54.5
Can use terminology and concepts in a given exercise or project	20	76.9	15	68.2
THE MEASUREMENT PROCESS				
Can remember the measurement process	14	53.8	13	59.1
Can use the measurement process in a given project/situation	20	76.9	12	54.5
TECHNIQUES AND TOOLS (T&T) FOR SOFTWARE MEASUREMENT				
Can remember the existing software measurement techniques and tools	17	65.4	13	59.1
Can give an example of software measurement techniques and tools	19	73.1	12	54.5
Can follow a technique in an exercise or project	20	76.9	10	45.5
SOFTWARE MANAGEMENT MEASURES				
Can remember concepts related to effort estimation, and measures for project planning and control	15	57.7	14	63.6
Can explain how to estimate the effort of a project	16	61.5	14	63.6
Can measure time and effort in a project	14	53.8	11	50.0
MEASURES FOR THE REQUIREMENTS PHASE				
Can remember the most common functional size measurement methods	14	53.8	12	54.5
Can obtain the functional size of the software in an exercise or project by following a measurement method	16	61.5	9	40.9
MEASURES FOR THE DESIGN PHASE				
Can remember the measures for the design phase	16	61.5	7	31.8
Can understand the meaning of the measures and how to obtain them	16	61.5	8	36.4
Can obtain the measures in an exercise or in a small project	16	61.5	8	36.4

Table 3.7 Verification of the ranking of the complementary skills

Skills	Strongly agree					
	Teachers			Practitioners		
	Ranking	#	%	Ranking	#	%
Critical Thinking	1	25	96.2	1	16	66.7
Oral Communication	2	13	50.0	4	12	50.0
Written Communication	3	16	61.5	3	11	45.8
Teamwork	4	16	61.5	2	14	58.3

As mentioned, the questionnaire for teachers included an extra section named Constructivism and Active Learning. The following tables contain the results of this section, which had three questions. The first question intended to discover preferences in methods for teaching software measurement. The results for this question appear in table 3.8, from which four preferences have been identified: class discussion, case studies, problem solving and games.

Table 3.8 Methods preferred for teaching software measurement

Methods preferred for teaching software measurement	Teachers =26	
	#	%
Class discussion	20	76.9
Role playing	8	30.8
Case studies	19	73.1
Games, simulations	14	53.8
Problem solving	19	73.1
Reflective journaling	5	19.2
Outdoor experience	2	7.7

The second question of this section aimed at identifying the resources that teachers consider as the most valuable to facilitate the implementation of active learning in their courses that include software measurement topics. Table 3.9 shows that three resources are considered very valuable for teachers: examples of software measurement; guidelines for applying active learning; and suggested activities to promote active learning in their courses.

Table 3.9 Resources for teaching software measurement

Valuable resources for teaching software measurement	Teachers = 26	
	#	%
Learning objectives	12	46.2%
Suggested content	11	42.3%
Guidelines for applying active learning	16	61.5%
Suggested activities to promote active learning	15	57.7%
Set of examples to be distributed to students	18	69.2%
Suggested assessment/evaluation criteria	11	42.3%

Finally, the third question asked teachers about the perceived impediments or barriers for adopting an active learning approach in their courses that include software measurement topics. Fifty percent of teachers consider that there are no impediments. The other fifty percent believes that the limited time they have to cover the course content, along with their own time constraints for preparing suitable activities as well as the lack of support and resources from universities are impediments (see table 3.10).

Interviews with recognized experts

Four software-measurement book authors were interviewed in October 2012. During the interview - 40 minutes in average, they were asked about their opinions with regard to the results obtained in the Delphi study. They had the freedom to make comments and give reasons that supported their agreement or disagreement with the Delphi's results.

To meet the ethics policies at ETS, an *information and consent* form was signed by the interviewees to assure them the confidentiality of the data collected (written notes and audio recordings).

Table 3.10 Impediments for adopting an active learning approach for teaching software measurement

Impediments for adopting an active learning approach	Teachers = 26	
	#	%
No impediments	13	50.0%
Limited time available for covering the course content	13	50.0%
Time constraints for preparing activities	12	46.2%
Lack of guidelines and resources for teaching using active learning	7	26.9%
Lack of guidelines and resources for assessing students' performance	3	11.5%
The training required to be able to apply active learning	3	11.5%
Students' aversion to the active learning approach (prefer a passive role)	0	0.0%
Active learning may not be suitable for teaching software measurement	0	0.0%
Lack of support and resources from university (labs, software, university-industry agreements, inflexibility for adopting new ways of teaching)	12	46.2%
Satisfied with the way I am teaching	3	11.5%

A semi structure type interview was used, which is characterized by (Bhamani Kajornboon, 2005; Leedy and Ormrod, 2010):

- having an interview guide (list of questions to be covered);
- flexibility: changing the order of or adding questions depending on the course of the interview;
- giving explanations and asking for clarification if the answer is not clear;
- using words that are considered best according to the situation;
- using a conversational style but keeping the focus on the guide.

The interview guide is available in Appendix XIII - interview section.

The results of the interview show that, in general, recognized experts agree with the results. However, some disagreements were observed which were explained through the interviewees' comments that are presented next.

Interviews with recognized experts: Ranking of software measurement topics

The software measurement topics kept the ranking order equal to the one obtained in the Delphi study (Basic Concepts 100% - i.e. the four experts agreed, Measurement process 75% - 3 experts agreed, Techniques and tools 100%, Software management measures 50%, measures for the requirements phase 75%, measures for the design phase 75%). One expert mentioned that software management measures should be taught after the specific measures (requirements and design measures). He commented that the current ranking of topics is logically ordered for trainees who work in organizations that already have historical data for performing estimations. He said that this is not the case of undergraduate students because they are in the process of learning software measurement, so they should learn first the measures for the requirements phase. Another expert ranked the topic software management measures as second since he considers that students need to know first why we need specific measures before performing estimations. Finally, another expert said that he agreed with the topics selected as priorities except for the *measures for the design phase* because he thinks it should not be a priority. According to him, the first five topics in the ranking are mandatory for university students.

Two experts suggested that the importance of software measurement should be strongly emphasized. One of them also recommended linking the importance of measurement with having clear objectives when measuring the software and the consequences of not doing that (measurement). In other words, students should be aware of potential lost of not estimating well for the lack of measurement. He suggested academia to look for ways of motivating students to measure.

Interviews with recognized experts: Ranking of skills

Among the four interviewees, two of the skills - presented for being ranked - reached consensus: Critical thinking (#1 - 100%) and written communication (#3 - 75%). The other

two skills did not reach a clear consensus. The explanations for making the ranking of the skills are presented next.

One expert said that there were 2 ways of ranking the skills. One is from an academic perspective and the other is in practical terms. According to him, from an academic view (for university students), the ranking should be: critical thinking, communication (oral and written) and team work. The practical ranking (for practitioners) may consider other skills to rank: leadership, problems resolution, ethical aspects, etc. Another expert mentioned that he would split critical thinking into 2 skills: critical thinking per se and problem solving. These two skills should be the firsts in the ranking, followed by oral communication, written communication and team work. Another opinion from an expert was to rank the skills taking into account that software measurement demands to work carefully. Therefore, he proposed the ranking as follows: critical thinking, writing skills, team work and oral skills. The last expert said that all those 4 skills selected by participants of the Delphi study are important. He suggested using the term *interpersonal* skills rather than *team work*. According to him, interpersonal skills are needed for convincing people about the value of measurement for organizations. Notwithstanding, he believes that the development of skills is difficult for young undergraduates.

Interviews with recognized experts: Levels of learning per topic

The opinions of the experts regarding the level of learning are somehow similar to the results obtained in the Delphi study and the surveys conducted at the verification phase. This means that the experts mostly chose the levels of learning that correspond to the first levels of the Bloom's taxonomy (remember, understand and apply). However, the following differences were observed:

In the case of the topic *measures for the design phase,* all experts agreed that students should only reach the two first levels of learning (remember and understand).

One expert considered that undergrads should not reach the *apply* level for the topics measurement process (use a measurement process in a project) and measurement techniques (follow a technique in an exercise or project). Another expert did not agree that students obtain the functional size of the software (apply). According to these experts (both with teaching experience), teachers may face time constraints while trying to deal with higher levels of learning (from apply to create). In addition, one of the experts said that the *measurement process* and the *techniques and tools* are context dependent. Hence, reaching understanding of these topics may be enough.

3.4 Interview with teachers

The objective of the interview was to obtain insights into the problems that teachers face in the teaching and learning process of software measurement.

Between October and November 2012, seven teachers were interviewed: four interviews took place in Assisi, Italy during the Joint Conference of the 22nd International Workshop on Software Measurement (IWSM-Mensura 2012) and the 7th International Conference on Software Measurement; and three were performed via Skype using only the audio facility. All of the interviewees had more than nine years of experience in teaching software measurement at university courses, and research experience as demonstrated through their publications in the field.

A semi-structured type interview was used. Participants were asked to sign an information and consent form to meet the policies of the ETS Ethics Committee (see the interview guide in Appendix XIV).

By talking to teachers, a number of problems related to the teaching of software measurement were identified, as follows:

- **Time constraints**: 5 out of 7 interviewees said that the number of hours assigned to teach software measurement into their courses is not enough to cover the program in depth. Teachers in charge of software engineering or project management courses for undergraduates mentioned that they hardly have 6 to 8 hours for covering software measurement concepts. Teachers, also mentioned that in the case of courses such as software quality, software project management, software metrics, systems management and planning, which have between 42 to 60 hours, covering in depth a whole range of topics (functional size measurement, effort estimation, estimation models, GQM, standards, product measures, PMBOK, etc.) is difficult. Four teachers said that they assign academic exercises/projects to students because there is no time for working with real projects.
- **Lack of resources at universities**: three interviewees mentioned the limitations they have in terms of: classrooms (not suitable workshops and group work activities); labs (not enough to have practical classes with students); many students per class (50 to 70); budget to increase or improve the relationship with industry partners, budget for research, etc.
- **Overloaded students**: Two teachers mentioned that some students at universities are or seem to be overloaded. According to teachers, these students give the minimum effort (just to pass the course), show a lack of motivation for learning new things, have aversion to do something else that demands more knowledge or work; and are reluctant to interact in class because of their lack of knowledge.
- **Students without pre-requisites**: One teacher expressed his worries of having students who do not have the pre-requisites necessary to learn software measurement (requirements elicitation, UML diagrams, and basic statistics). This means that, teachers have to spend extra time to explain some concepts needed to learn software measurement.
- **Social issues**: One teacher mentioned that the differences among students coming from a variety of countries with dissimilar cultures, languages and backgrounds are barriers that make the interrelation among teacher-learners, learners-learners difficult, and lead the students to keep a passive role in class.

When teachers were asked about how they determine the level of learning that students reach in software measurement (e.g remembering terminology, obtaining the functional size of the software in an exercise, using a technique in a project, etc) and if they experience problems in doing so, teachers gave the following answers:

- **I do not determine the level of learning** (2 teachers). According to their explanations, determining the level of learning of each student takes time. Moreover, one of them explained that- for him- it is also difficult to interpret what students have really learned by reading answers in the exams.
- **Rubrics help me to identify the level of learning** (3 teachers). One teacher said that the rubric used to grade the projects was in his head since he has been teaching the course for a long period. The other two teachers said that their rubrics were developed to identify the levels of learning reached by students. One teacher said that he gives the rubric to students in advance - before grading the assignment.
- **The exam questions are designed to test the level of learning reached by students** (3 teachers). Teachers mentioned that they get an idea of what students learned based on the responses provided in the exam. One teacher said that he uses multiple choice or True/False questions to warm up students, but he uses open questions to know the level of understanding of students. Other teacher said that he only uses open questions because when the students are nervous they fail multiple choice questions; therefore, open questions allow student to freely express their ideas and understanding.
- **Projects help to determine what students learned** (4 teachers). One teacher explained that in his case, the project is optional. Hence, for him it is easier to see how much students learned through the project rather than from the written exam. Other teachers mentioned that according to the completeness, quality and consistency of the project, they get an idea of the knowledge acquired by students.
- **Anything is used to determine the level of learning** (1 teacher). In this particular case, the teacher mentioned that in his university there are policies regarding the level of learning reached by students. This means that students have to demonstrate throughout individual work (written exam, oral presentations, and assignments) and a group project what they have learned. Everything counts for determining if the students reached the

passing levels (minimum level of learning expected in the course, excluding memorization).

During the interview, teachers were asked to indicate among a list of resources, the ones considered as the most valuable to teach software measurement with an active learning approach. The answers were: examples to be distributed to students (7), suggested activities to promote active learning (5), suggested assessment criteria (4), guidelines for applying active learning (3), ideas of learning objectives associated with the software measurement topics (3).

Finally, regarding the topics that teachers consider as essential to be taught for undergraduates, their suggestions can be summarized as follows:

- Basic concepts of software measurement: units, scales and basic statistics.
- Measurement process: the very basis of the measurement process because undergrads lack of experience to understand the management aspects and decision making process.
- Techniques and tools: techniques rather than tools because tools are difficult to obtain at universities while techniques are of easy access (GQM, PSP, root-cause analysis, etc).
- Software management measures: estimation of duration and effort; and the discipline in performing such measures for project management purposes.
- Functional size measurement: identification of basic functional components.

3.5 Consolidation of results

The results of the studies presented in the previous sections have been consolidated in Figure 3.2, which is a layer representation of the software measurement topics that should be covered in universities. The innermost layer is the most important or essential when teaching software measurement; the second layer is the second most important and so on. The bold letters correspond to the topics that are considered as priorities in software measurement education for undergraduate programs (our target). The remaining topics - written in light

gray and located in the upper layers - are the topics that may be covered superficially for undergrads or taught in graduate programs.

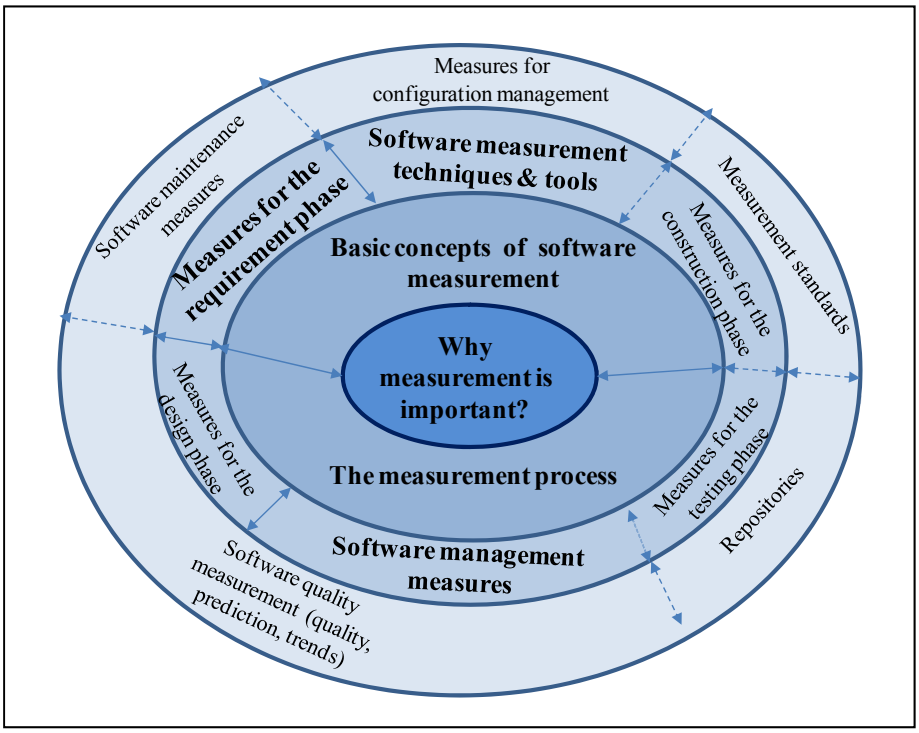

Figure 3.2 Layers of the software measurement topics

In summary, five out of thirteen topics (see Figure 3.3) have been identified as priorities in software measurement education for undergraduate students. For these topics, the levels of learning suggested to reach are the ones that fall in the first three levels of the Bloom's taxonomy (remember, understand and apply). In addition, educators should take into consideration that four skills are needed to complement the education of software measurement in students: critical thinking, oral and written communication, and team work. All of these findings have been considered for developing the educational framework presented in the next chapter.

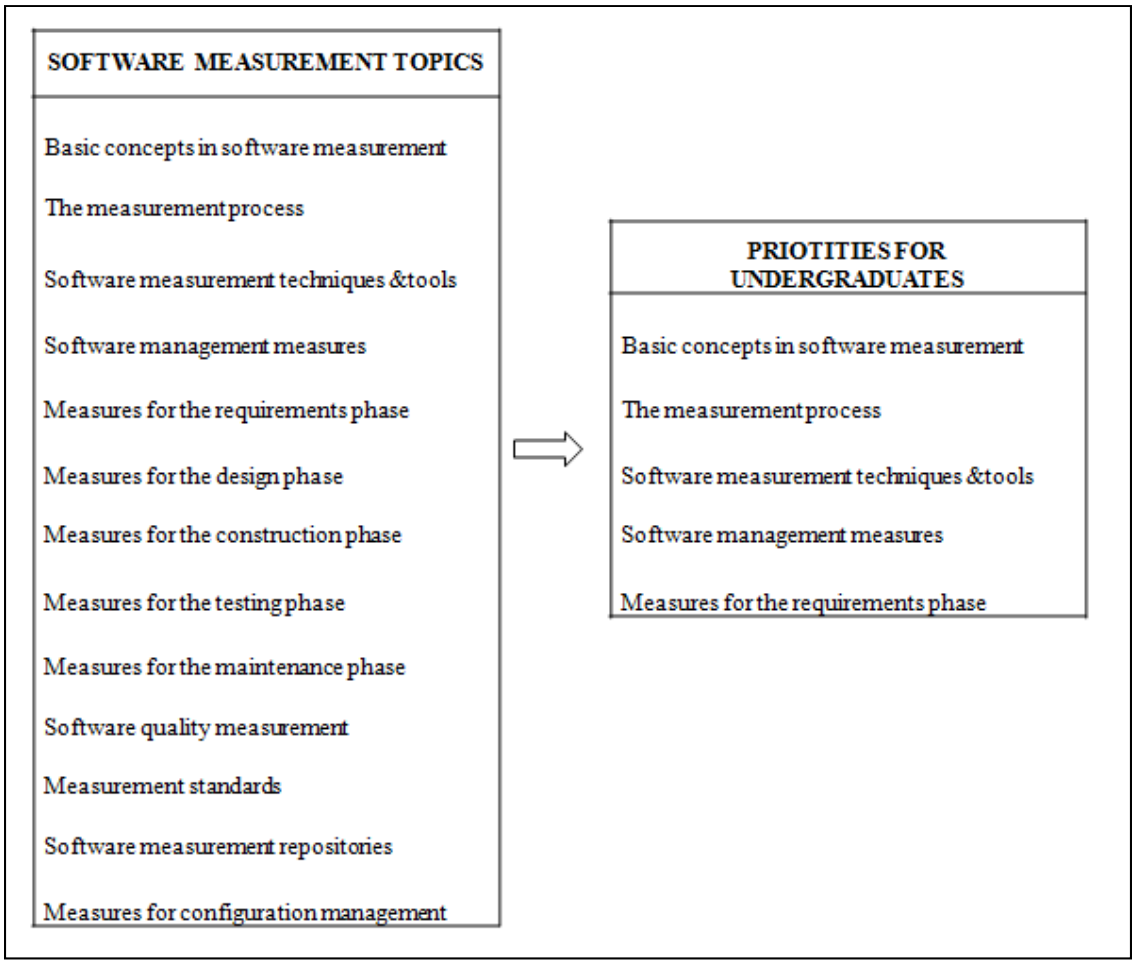

Figure 3.3 Software measurement topics considered as priorities in software engineering education for undergraduates

CHAPTER 4

A FRAMEWORK FOR THE EDUCATION OF SOFTWARE MEASUREMENT

This chapter presents the definition and structure of an educational framework developed to facilitate the teaching and learning of software measurement. To illustrate the applicability of the framework, five examples have been designed - one in this chapter and four in Appendix XV. Each example refers to the priority topics and learning outcomes identified by experts in the field through the Delphi study explained in chapter 3. The examples contain guidelines, activities, tasks and rubrics that address the reaching of learning outcomes in students.

4.1 Framework design

This framework is designed as a set of guidelines to assist university teachers and instructors in the teaching and learning process of software measurement for software engineering undergraduate students or beginners in the field. This framework was developed on the basis of:

- the related bodies of knowledge (Abran, April and Buglione, 2010);
- the results of the surveys and the Delphi study -chapter 3;
- the Bloom's taxonomy on levels of learning (revised version (Anderson et al., 2001)); and
- the constructivist approach (Brooks and Brooks, 2001; Fosnot, 2005).

4.2 Objective of the framework

The objective of the framework is to provide guidelines to university teachers and instructors in order to promote the achievement of learning outcomes in students that are learning software measurement for the first time. This way, the framework aims to be an instrument for education in the software measurement field to enhance the teaching practices at the university level. This improvement is in terms of new alternatives that teachers can follow to improve their teaching.

The framework can be used as a starting point to teach software measurement as part of a software engineering course, or any other courses for beginners in which topics related to software measurement are covered.

4.3 Structure of the framework

The framework has three main components: the inputs, the guidelines for teaching and learning, and the expected outcomes (see Figure 4.1).

The set of inputs consists of:

- the software measurement topics shown in Figure 3.3;
- the course's objectives; and
- the available resources for teaching and learning (i.e. the Bloom's taxonomy; teaching, learning and assessment approaches; bodies of knowledge; software measurement books, etc).

The guidelines are divided into two parts: content and constructivist approach - see Figure 4.1. The former relates to each of the priority topics from Figure 3.3 along with the identification of their cores (i.e. the most essential part to be covered in depth during the course sessions). The cores for the topics were identified through interviews with experienced teachers and highly recognized experts in the field, as explained in chapter 3. The latter includes a set of activities and tasks designed to facilitate the teaching and learning process of software measurement topics. In addition, adequate feedback to students is suggested in order to ease the achievement of the expected learning outcomes.

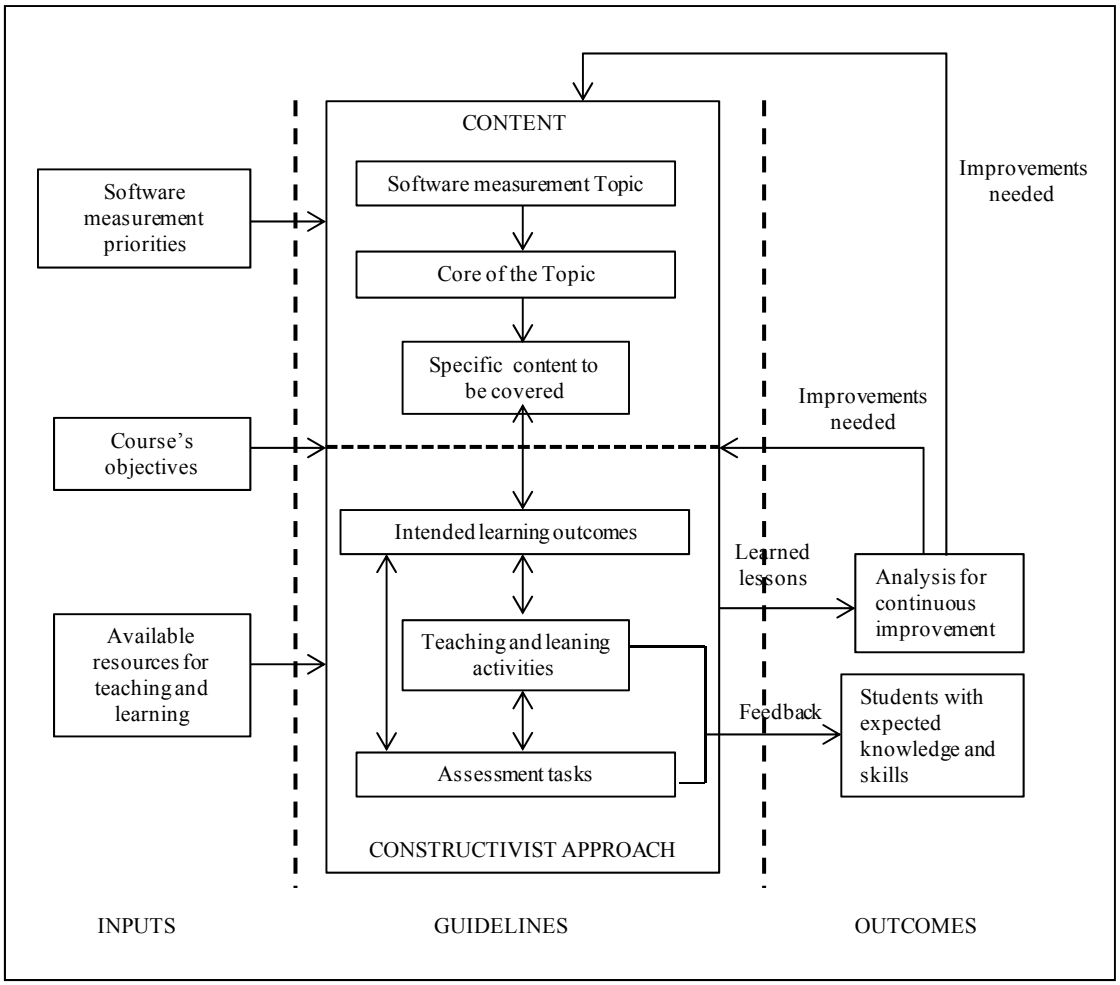

Figure 4.1 Structure of the educational framework

The expected outcomes refer to the knowledge and complementary skills that students are expected to develop. In addition, an analysis is advised to determine the extent to which the guidelines provided in the framework are contributing to reach the desired outcomes. The analysis will investigate how well the students achieved the expected outcomes, and the appraisal of unplanned course of actions and their endings. The analysis is very important to improve the guidelines and promote better teaching. Notwithstanding, as expressed by Hagström and Lindberg, "educational processes can never be completely prescribed" due to the flux of teachers, students, knowledge, context, learning goals, etc. (Hagström and Lindberg, 2012).

4.4 Objectives of the software measurement course for beginners

This course aims to familiarize the audience (undergraduate students or beginners) with basic terminology, concepts and methods commonly used in the software measurement domain. At the course completion, the students will be aware of the importance of measuring the software (product and processes). Also, they will be able to apply basic software measurement knowledge and a technique to measure (for example: functional size, time and effort) a small-well-documented set of simple functional requirements. In addition, the course promotes the development of interpersonal, communications and thinking skills by exposing the students to a number of individual and group activities.

4.5 Pre-requisites

To use this framework, previous knowledge of software measurement is not necessary. However, the students are expected to have basic knowledge of requirements specifications and associated diagrams (e.g. working with use cases, class and sequence diagrams).

4.6 Applicability of the framework

This section exemplifies the application of the framework through the development of one of the topics: Measures for the requirements phase. This topic was selected - among the five priority topics - to be explained in this chapter because it is one of the most demanding in terms of activities and tasks required to reach the learning outcomes. The other four topics are presented in Appendix XV.

For each topic, the framework suggests the content to be covered, the teaching and learning activities (TLAs), and the assessment tasks (ATs) that promote the acquisition of knowledge and the achievement of the intended learning outcomes (ILOs). In addition, a roadmap per topic was designed to illustrate their guidelines. The presentation of every topic, in this chapter and Appendix XV, is divided into four parts: content, intended learning outcomes, teaching and learning activities, and assessment tasks. When needed, each part contains

examples and bibliographic references. Table 4.1 presents the general view of the whole content of the framework.

Table 4.1 The educational framework at a glance

Topics	Intended Learning Outcomes (ILO)	Teaching and Learning Activities (TLA)	Time (min)	Assessment Tasks (AT)
1. Basic concepts of software measurement	1.1 Explain why measurement is important. 1.2 Give examples of measurement units and scale types. 1.3 Use measurement units and scale types.	1.1 Interactive lecture with two activities	60	1.1 Questions in the mid-term exam 1.2 Group project: be familiar with terminology used in the ISBSG questionnaire
2. The measurement process	2.1 Follow a given measurement process in a group project	2.1 Lecture including an example of a measurement process	30	5.3 Group project: Follow a measurement process
3. Software measurement techniques	3.1 Give examples of techniques for software measurement 3.2 Follow a given technique in a group project	3.1 Interactive lecture with one activity in pairs	90	3.1 Quiz 5.3 Group project: Use a technique
4. Software management measures	4.1 Measure duration and effort in a group project	4.1 Interactive lecture with examples	30	5.3 Group project: Duration and effort, reflection about the results
5. Measure for the requirements phase	5.1 Explain how the functional size measurement (FSM) methods work. 5.2 Obtain the functional size of a small-well-documented set of simple functional requirements	5.1 Reading prior to class 5.2 Lecture including 2 examples of FSM 5.3 Group activity in class	90	5.1 Short essay in class 5.2 Open questions in the final exam 5.3 Group project

4.6.1 Example: Measures for the requirements phase

Figure 4.2 shows the roadmap of the guidelines suggested for the topic *measures for the requirements phase*. In the roadmap, the core of the topic and the content are defined. Therefore, this section only explains where to find the suggested content and how to perform the activities that promote the achievement of learning outcomes in students.

4.6.1.1 Suggested content

The content refers to the subjects that are suggested to be covered in each of the topics included in this framework.

What is Functional size?

Functional size is defined in ISO 14143-1:2007(E) as: "a size of the software derived by quantifying the Functional User Requirements (FUR)," where FUR is as "a sub-set of the User Requirements describing what the software shall do, in terms of tasks and services" (ISO/IEC, 2007).

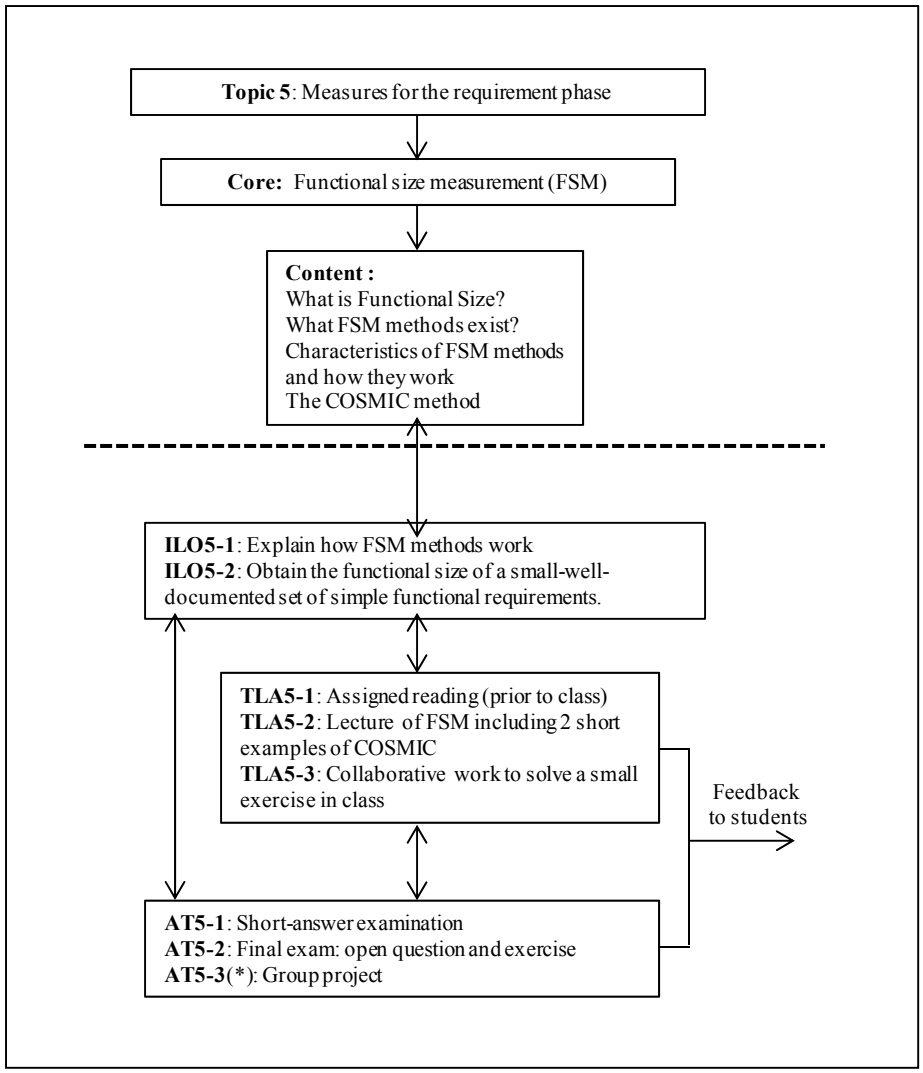

Figure 4.2 Example - Software measures for the requirements phase

Examples of FUR are:

- input of students data in a registration system;
- calculate the average mark of students in a course;
- list the students that are above the average.

More examples of FUR can be found in (ISO/IEC, 2007).

Functional size measurement methods (FSM)

"Functional Size Measurement (FSM) is a technique used to measure the size of software by quantifying the Functional User Requirements of the software" (ISO/IEC, 2007).
"Functional Size Measurement Method (FSMM) is a specific implementation of FSM defined by a set of rules, which conforms to the mandatory features of ISO/IEC 14143-1:2007" (ISO/IEC, 2007)

In 1979, Allan Albrecht published the first functional size measurement method known as Function Point Analysis. Afterwards, several extensions and variations of this method have been produced. The current functional size measurement (FSM) methods adopted as ISO standards are (Abran, 2010; Bundschuh and Dekkers, 2008; Fetcke, 1999):

- ISO 20926:2009 IFPUG 4.1 functional size measurement method
- ISO 19761:2011 COSMIC functional size measurement method
- ISO 24570:2005 NESMA functional size measurement method
- ISO 20968:2002 MKII function point analysis
- ISO 29881:2010 FiSMA 1.1 functional size measurement method

The first two ISO FSM methods are the most commonly used (Jones, 2008; Villavicencio and Abran, 2011b).

Characteristics of the functional size measurement methods

The following mandatory characteristics of the FSM methods are listed in the standard ISO 14143-1:2007(E) (ISO/IEC, 2007).

A FSM method is:

- independent of the methods used to develop the software being measured;
- independent of the methods used to support the software being measured;

- independent of the technological components of the software being measured.

This implies that functional size is not:

- derived from the effort required to develop the software being measured;
- derived from the effort required to support the software being measured;

How the functional size measurement methods (FSMM) work

All FSMM must fulfill the mandatory requirements of ISO/IEC 14143-1: this means that regardless of the rules of the measurement methods, all these methods must focus on measuring the functional user requirements.

Based on the standard ISO/IEC 14143-1, a FSMM "shall include the following activities in order to derive Functional Size:

- Determine the Scope of the FSM (purpose for measuring the software);
- Identify the Functional User Requirements within the Scope of the FSM;
- Identify the Basic Functional Components (BFC) within the Functional User Requirements (A BFC is an elementary unit of Functional User Requirements defined by and used by an FSM Method for measurement purpose)
- Classify BFCs into BFC Types, if applicable;
- Assign the appropriate numeric value to each BFC;
- Calculate Functional Size".

This means that the Functional User Requirements (FUR) are characterized in terms of Basic Functional Components (BFC).

For example, the BFC for the COSMIC Functional Size Measurement Method is the data movement, categorized into four BFC types: Entry (E), Exit (X), Read (R), and Write (W). In the case of the IFPUG functional size measurement method, the BFC types are: External Input (EI), External Output (EO), External Inquiry (EQ), Internal Logical File (ILF), and

External Interface File (EIF). These five elements are the BFCs for this method. Figure 4.3 shows a general representation of the COSMIC functional size measurement method ISO 19761.

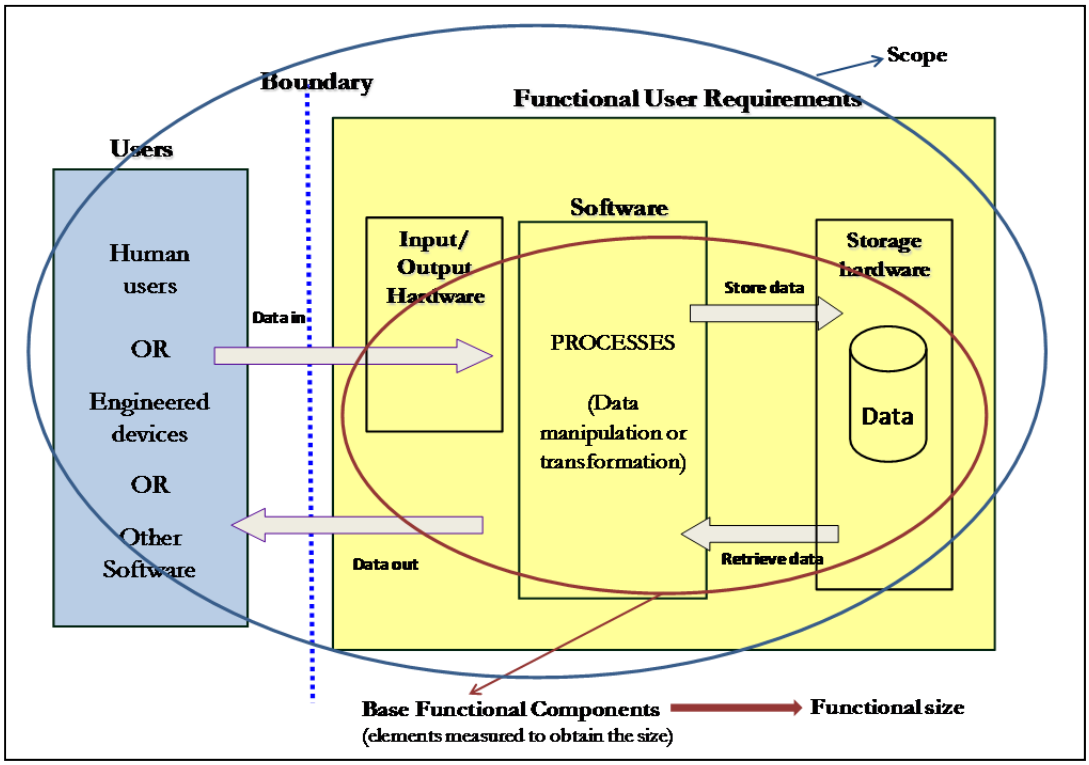

Figure 4.3 General representation of the COSMIC Functional Size Measurement Method

Available material for teaching this subject can be found in the standard ISO 14143-1:2007(ISO/IEC, 2007)

The COSMIC method

All the information related to this method is freely available throughout its website http://www.cosmicon.com/. The site allows visitors not only having an overview of the method, but downloading related articles, case studies, and the most recent version of the measurement manual translated in 12 languages. In addition, the site contains a Frequently Asked Question section useful for all type of audiences.

A brief overview of the COSMIC method is as follows:

1) The Basic Functional Component (BFC) is the data movement.
2) The measurement unit is a COSMIC Function Point (CFP), which represents one data movement of one data group.
3) A functional user of a software application can be: a human, another software application, or a hardware device.
4) A boundary is a conceptual interface between the functional users and the software application.
5) The functional users interact with the software application via data movements.
6) There are four types of data movements: Entry (E), eXit (X), Read (R), and Write (W).
    a) An *Entry* occurs when a data group is moved from a functional user into the software.
    b) An *eXit* occurs when a data group is moved from the software to a functional user.
    c) A *Read* occurs when a data group is moved from a persistent storage into the software.
    d) A *Write* occurs when a data group is moved from the software into a persistent storage.

A *functional process* is a set of data movements (at least 2).
A *functional process* is an elementary component of a set of *Functional User Requirements*.
A functional process *is triggered* by a data movement (an Entry) from a functional user.
A data movement moves a single *data group,* which is a distinct, non-empty, non-ordered and non-redundant set of *data attributes* of the same *object of interest*.
A size unit of 1 CFP (COSMIC Function Point) is assigned to any data movement
The *functional size* of software is calculated in CFP by adding together the data movements.

To illustrate the above steps and concepts, Figure 4.4 shows a graphical representation of how to obtain the functional size of a functional user requirement by using the COSMIC method. The explanation of this Figure is next.

Scope of measurement: Measure the size of the functionality "Create a new customer"

Functional user: Salesman

Pre-conditions: The salesman is already logged in the system and has selected the option "Create Customer"

Flow of events:

1) The salesman enters the name and email of the new customer (John Smith, js@hotmail.com) and presses OK.
2) The system verifies if the customer already exists in the database.
3) If the customer exists, an error message is displayed.
4) If the customer is new, the system asks the salesman to confirm the data that is going to be saved by pressing OK. If needed, the salesman can correct the customer data.
5) A new customer is created into the database.
6) A confirmation or error message is displayed.

Boundary: Limit between the salesman and the system

Functional processes:

1) Verify if the customer exists in the database,
2) Create a new customer in the database.

Data group (DG): Customer (Name & e-mail)

Data attributes: Name, email

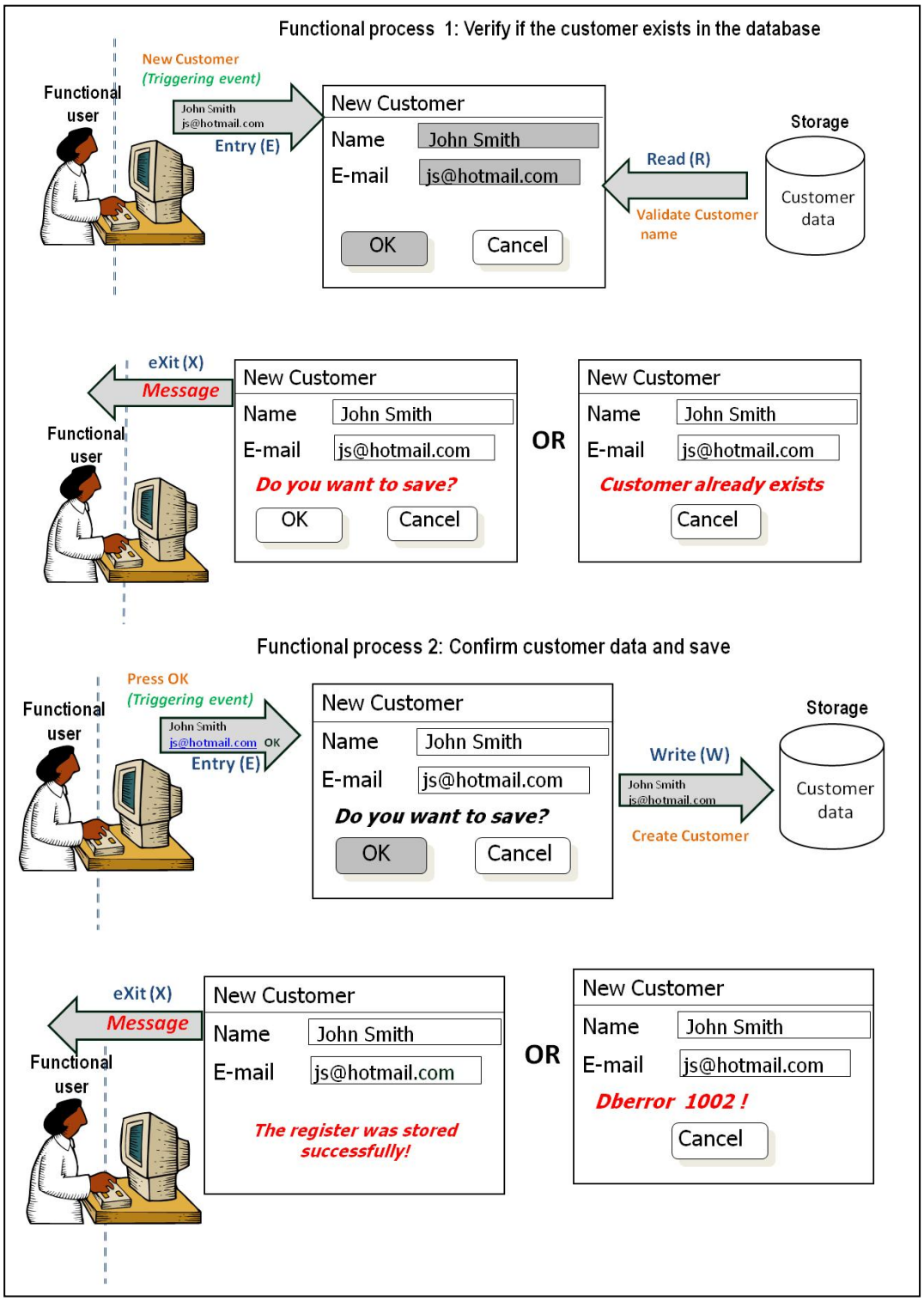

Figure 4.4 Example of the COSMIC method

Data movements (DM):

Functional process 1: Verify if the customer exists in the database

- The salesman enters the Customer data and presses OK (Triggering event - **E**ntry)
- Retrieve the existing customers from the database to verify if John Smith already exists as customer (**R**ead).
- A confirmation message OR an error message -if the customer already exists- (e**X**it).

Functional process 2: Create a new customer in the database.

- The salesman verifies/corrects the customer data and presses OK (Triggering event - **E**ntry)
- The customer data is stored in the database (**W**rite)
- Confirmation OR Error message (e**X**it)

Functional size:

Assign 1 CFP (Cosmic Function Point) per Data Movement, as shown in Table 4.2.

The examples of this chapter and other examples of functional size measurement are available in http://software-measurement-education.espol.edu.ec/ (see also Appendices XV and XXVII - a COSMIC case study).

All the examples of this chapter and

Suggested references for beginners are:

- The Warehouse Software Portfolio, A Case Study in Functional Size Measurement by T. Fetcke 1999, chapter 7.
- Overview of the COSMIC Functional Size Measurement Method, available in: http://www.cosmicon.com/methodV3.asp.

- COSMIC Measurement Manual, available in: http://www.cosmicon.com/portal/dl_info.asp?id=73, chapters 1 and 4.
- Case study - Course registration system, available in: http://www.cosmicon.com/dl_manager2.asp?cat_id=68&parent_id=17&parent_name=04+-+Case+Studies&sub_name=Business

Table 4.2 Example of counting Function Points with the COSMIC method

Functional Process	Subprocess	DG	DM	CFP	Comment
Verify if the customer exists in the database	The salesman enters the Customer data and presses OK	Customer	E	1	
	The system retrieves the existing customers from the database to verify if John Smith already exists as customer	Customer	R	1	
	A confirmation message OR an error message (if the customer already exists)	Software Messages	X	1	
Create a new customer in the database	The salesman verifies/corrects the customer data and presses OK	Customer	E	1	This movement is considered as an Entry because the salesman can retype the customer data
	The customer data is stored in the database	Customer	W	1	
	Confirmation or error message	Customer	X	1	

Total functional size = 6 CFP

4.6.1.2 Intended Learning Outcomes

For this example, two Intended Learning Outcomes (ILOs) are proposed. For each ILO, the corresponding level of learning according to the Bloom's taxonomy is in parenthesis, as follows:

- ILO**5**-1: Explain how the functional size measurement methods work (Understand).
- ILO**5**-2: Obtain the functional size of a small-well-documented simple set of functional requirements (Apply).

The number **5** in the ILO identifies the ordering of the topic according to the priority determined in the Delphi study (1: Basic concepts of software measurement, 2: The measurement process, 3: Software measurement techniques, 4: Software management measures and **5**: Measures for the requirements phase).

4.6.1.3 Teaching and Learning Activities

To reach the two ILOs (ILO5-1 and ILO5-2), TLAs (Teaching and Learning Activities) and ATs (Assessment Tasks) were selected by having in mind the type of knowledge - declarative or functioning - that students should reach. Biggs defines declarative knowledge as something that a person knows about (i.e. knowing "what"), and functioning knowledge as putting declarative knowledge to work by solving problems, designing, etc. (i.e. knowing "how" and "when") (Biggs and Tang, 2007). Other authors refers this latter as procedural knowledge (Anderson et al., 2001). ILO5-1 and ILO5-2 correspond to the functional - procedural- knowledge type. Figure 4.5 illustrates some activities and tasks that may engage students in their own learning (active learning) in order to achieve those ILOs.

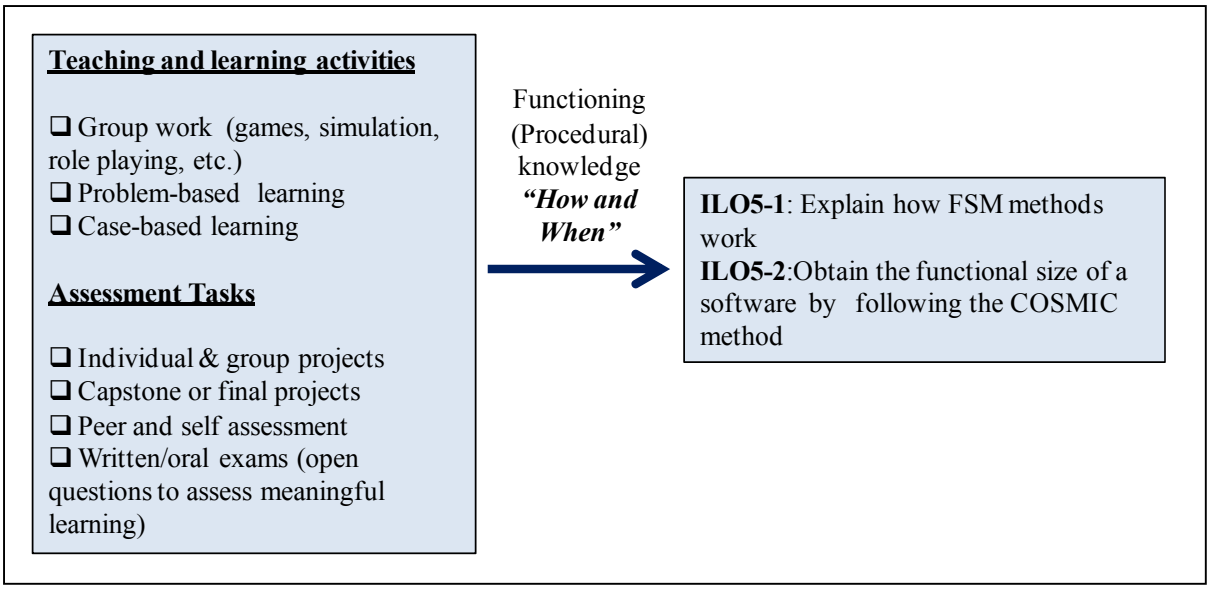

Figure 4.5 AlternativesTeaching and Learning Activities (TLAs) and Assessment Tasks (ATs) to reach the Expected Learning Outcomes (ILOs)

Regarding the activities used in class for promoting active learning, Beard and Wilson created the following four-stage sequence or activity wave based on the Cornell's flow of learning (1989) (Beard and Wilson, 2006):

1) Stimulate the learner enthusiasm by using ice-breakers.
2) Start to focus attention on what should be learned with medium-sized activities and narrow skills such as listening or questioning.
3) Direct the learner experience with larger activities and broader skills such as communication, team work, problem solving and critical thinking.
4) Share learner enthusiasm using regular reviewing activities - feedback.

Table 4.3 shows the link between the stages of the flow learning with the activities and tasks developed for the current example.

Table 4.3 Application of the Cornell's flow learning

Stages of the Flow Learning	Activities
Stimulate	Use of ice breakers
Focus attention	TLA5-1, TLA5-2, AT5-1
Learner experience	TLA5-3, AT5-2, AT5-3
Feedback	TLA5-3, AT5-3

In addition, for achieving the learning outcomes, a constructive alignment -represented as two-way connections arrows in figure 4.2- is proposed. Constructive alignment means that learners can construct knowledge when the Teaching and Learning Activities (TLAs) promote the ILOs and when the Assessment Tasks (ATs) are used to verify the ILOs level of achievement (Biggs and Tang, 2007).

TLA5-1: Reading prior class

Assigning a reading before a class session may reduce the high dependency on the lecture. By reading, students have the chance to reflect which, in turn, may enable them to reach a higher level of cognitive processing (Garrison and Archer, 2000). To take advantage of a reading, it is necessary to make students aware of: why the reading is important, how it relates to classroom activities and what information they should look for (Meyers & Jones, 1993 as referred in (Garrison and Archer, 2000)). This implies that teachers/instructors must provide students with guiding questions upfront in order to facilitate the remembering and understanding of the FSM concepts and application. Example of guiding questions for the reading could be:

- What can be measured with functional size measurement methods?
- What cannot be measured with functional size measurement methods?
- What is a Functional User Requirement (FUR)?
- From what kind of software artifacts can the FURs be obtained to measure the functional size of the software?

An example of a short reading that includes answers for the guiding questions is *The Introduction of the COSMIC Method* from the document: COSMIC Method v3.0.1 Measurement Manual pg. 10-14 (COSMIC, 2009).

TLA5-2: Lecture of FSM with examples

The suggested TLA5-2 activity is a traditional lecture - 60 minutes maximum. Lectures are useful for introducing a subject; however, they must be limited to cover few topics and to have, at the end, some time to review what has been learned. The lecturer has to keep in mind that the attention of the students drops after 15 minutes (Biggs and Tang, 2007; Garrison and Archer, 2000). That is why, changes of activities, use of icebreakers or pauses are required (e.g. ask for questions, ask questions, give 2 minutes break, use a 3 minutes icebreaker, etc). Using ice breakers -before starting or during the lecture - could be positive to students for reducing inhibitions, encouraging cooperation, developing social skills, creating trust, empathy and teamwork (Beard and Wilson, 2006; Knox, 2009). Information and examples of ice breakers can be found in (Knox, 2009).

In order to reach the two ILOs and to effectively manage time constraints, the focus of the lecture should be on: what is FSM, its characteristics and how to measure. Time constraints are an issue for teachers who usually try to cover as much material as they can; nevertheless, this should be avoided. It is better to ensure the learning of the essentials instead of superficially covering the whole study program (Biggs and Tang, 2007). A good way of assuring the students' learning is through examples that connect students with the surrounding world.

The slides designed by Abran 2011 which contain two short examples of measuring FSM using COSMIC, could be used for this lecture (see Figures 4.6 and 4.7) (Abran, 2011):

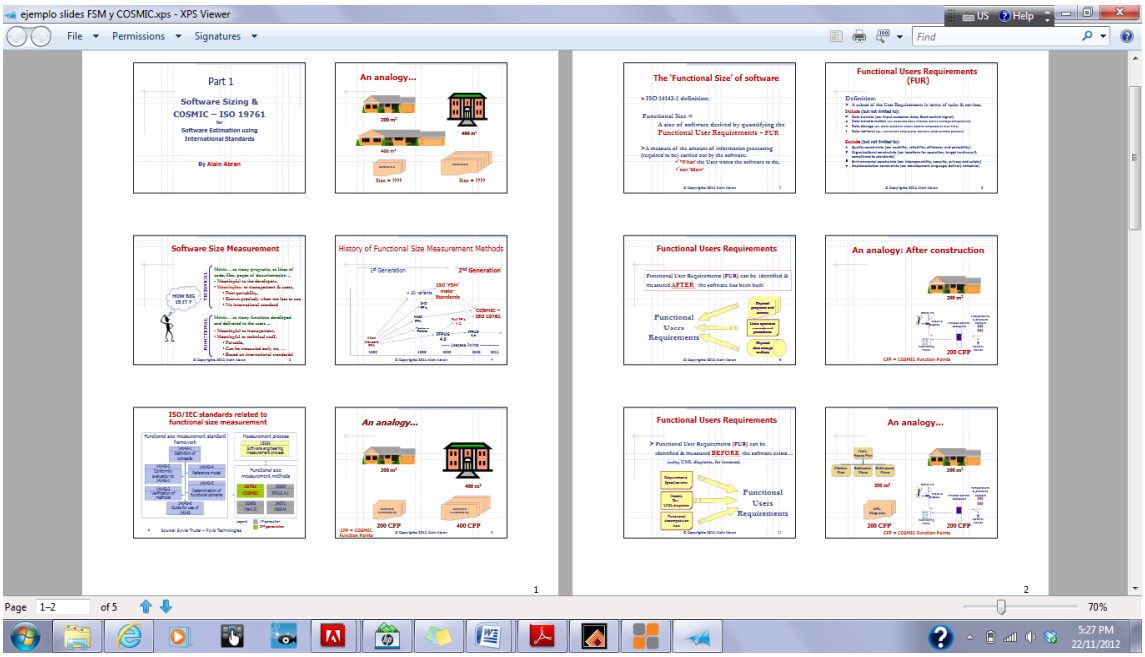

Figure 4.6 Example of the slides for a lecture of FSM - part 1

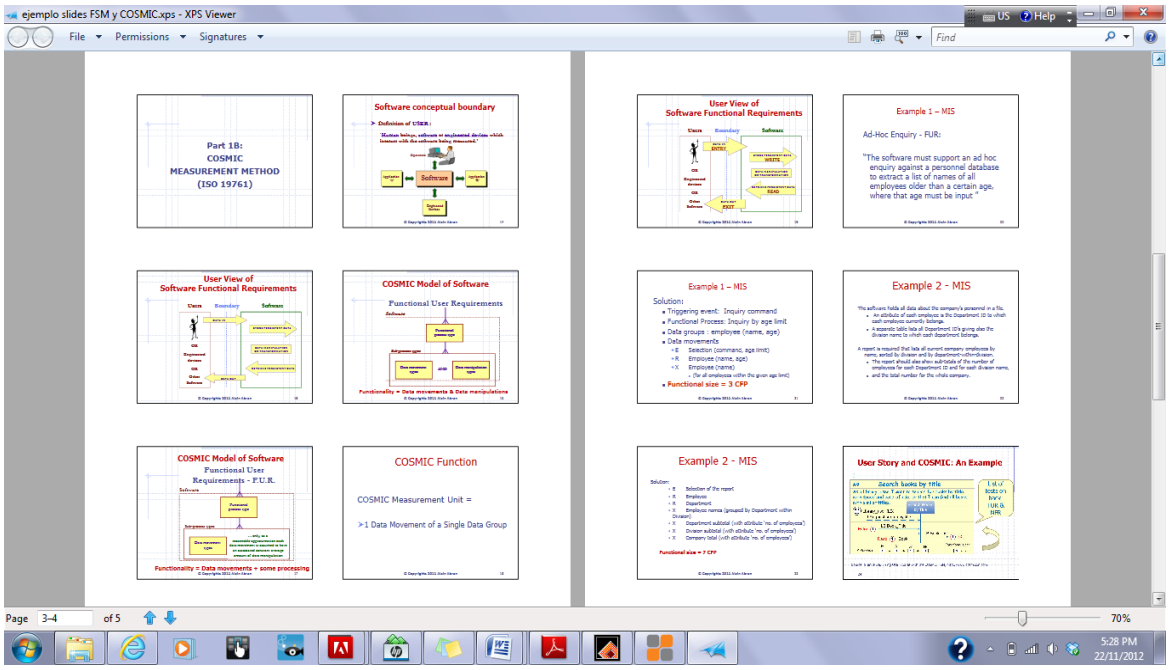

Figure 4.7 Example of the slides for a lecture of FSM - part 2

Depending on the availability of time, teachers can make use of games or other educational resources for achieving the learning goals (see examples in Appendix XV: Basic Concepts in Software Measurement - section 3).

TLA5-3: Collaborative work

The third proposed activity (TLA5-3) is a collaborative work to solve a problem in class that intends to facilitate the reaching of a higher level of learning (APPLY). Working collaboratively has several advantages for students who usually need to "share concerns, seek clarification over assignment requirements and to check their own insecure interpretations" (Saberton 1985 as referred in (Biggs and Tang, 2007)). By solving a problem, students put in practice what they have read (TLA5-1) and listened to (TLA5-2). Hence, the learning is fostered through the students' engagement along the collaborative and guided activity for solving a problem - an exercise in class.

To perform this activity, the following steps may be taken into consideration:

- Prepare a short exercise of functional size measurement (e.g. a purchase order, a hotel booking, a student registration, etc.). Take into account that exercises/examples must be in the context in which the students will use the problem solving skills (i.e. real-life problems).
- Form small groups of students to perform the exercise - games may be used.
- Give students detailed instructions including a reasonable timeframe for performing the activity.
- Select 2 groups to present the solution.
- Ask questions to the students and invite the rest of students to present different solutions
- Make suggestions and give feedback in a critical and reflective way to demonstrate the correct answers to students. In this way students are shown how to think critically. Therefore, this demonstration will help students in developing analytical skills.
- Encourage the learner to reflect on what he/she has learned (self reflection).

- Distribute the solution of the exercise to students including explanations.

Regarding step 2 above, some suggestions of how to create groups are available in (Examiner, 2012).

For this TLA5-3, a simplified version of an exercise of a "purchase order" designed by Trudel 2012 (Trudel, 2012) is a good example that can be distributed to the students. This example includes a description of the flow of events, a screen shot of the user interface and the data model (see Figure 4.8). With this information, along with an explanation from the teacher, the students have to obtain the functional size of the "purchase order" functionality in 20 minutes by using the following measurement procedure:

5) Identify the functional users
6) Identify the triggering event
7) Identify the functional processes
8) Identify the data groups
9) Identify the data movements from the interface (**E**ntry, e**X**it, **R**ead and **W**rite)
10) Obtain the total number of Cosmic Function Points (CFP)

The details and solution of this example are available in appendix XV.

All the elements included in the example (flow of events, screen shot, data model, measurement procedure) are essential to show students how to solve problems by using the COSMIC method. Structured methods are necessary to teach problem-solving skills (The Centre for Teaching Excellence, 2013).

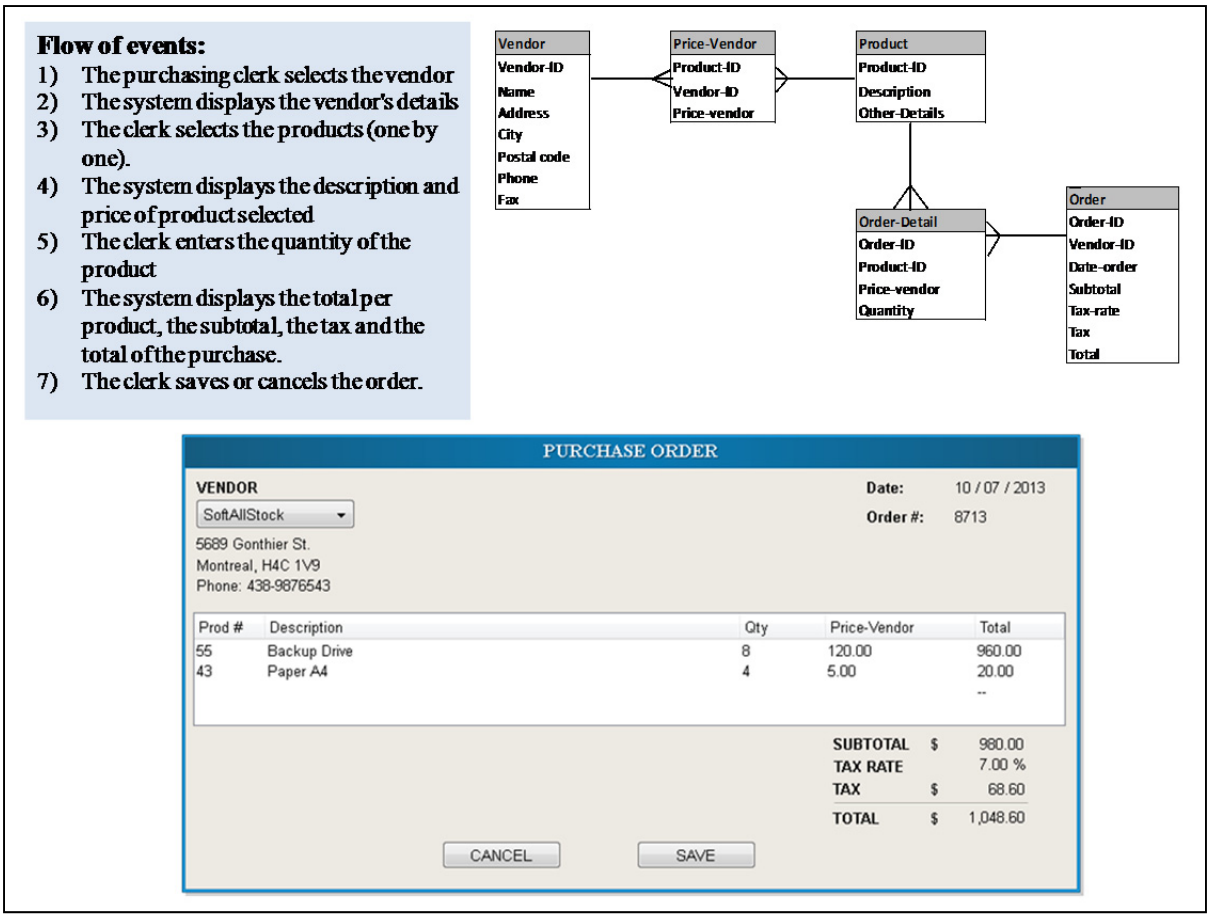

Figure 4.8 Example of an exercise of FSM

4.6.1.4 Assessment Tasks

The way students learn mainly depends on how they think they will be assessed (Biggs and Tang, 2007; Garrison and Archer, 2000). Therefore, the assessment has a very powerful effect on a student's approach to learning (Garrison and Archer, 2000). In this respect, if teachers want to assess deep learning instead of surface learning, they should communicate to students their expectations in advance. In addition, teachers must design assignments suitable for reaching deep learning and for focusing students on the most important concepts and skills that they have to acquire (learning outcomes). To focus students, teachers should provide good prompts -clear instructions and guidance on what students have to do (see chapter 1).

Since it is important to grade students (summative assessment - see chapter 1), a good way to do it is through rubrics. A rubric is a scoring guide that describes the criteria that will be used to evaluate student assignments (Biggs and Tang, 2007; Suskie, 2009). Some advantages of using rubrics are quoted by Suskie 2009 (Suskie, 2009), as follows:

Rubrics..

- Help clarify vague, fuzzy ILOs
- Help students understand the teacher's expectations
- Can help students self-improve
- Make scoring easier and faster
- Make scoring more accurate, unbiased and consistent
- Improve feedback to students
- Reduce arguments with students

In addition, Suskie (Suskie, 2009) give tips for creating effective rubrics, such as:

- Look for models or examples
- List the things you are looking for:
    i. What do you want students to learn from the assignment (ILOs)?
    ii. What are the skills do you want students to demonstrate in the assignment?
    iii. What are the characteristics of good student work?
    iv. What specific characteristics do you want to see in completed assignments?
- Leave room for the unexpected (encourage originality/creativity)
- Create a rating scale - at least 3 levels:
    i. Excellent, very good, adequate, needs attention
    ii. Strongly agree, agree, disagree, strongly disagree
    iii. Complete evidence, partial evidence, minimal evidence, no evidence
    iv. A, B, C, D, F

Rubrics are very useful to identify areas in which learners need to improve. This identification can be performed by teachers or students. If a rubric is distributed to the learners along with a task or if the learners are asked to apply a rubric to their or classmates task, they (the learners) might identify by themselves opportunities for improvements. This information (improvements needed) is useful for teachers to guide students in where to go next (feed forward). That is, giving to students ideas, guidelines or strategies to move forward in their understanding (Hattie and Timperley, 2007).

Examples of assessment tasks, rubrics and prompts for assessing ILO5-1 and ILO5-2 are presented next.

AT5-1: Assessing reading comprehension

The task AT5-1 is designed to assess how well students remember and understand the concepts of FSM (ILO5-1) after the assigned reading (TLA5-1). In this respect, a short essay (1 to 3 minutes) with one of the following questions could be used:

Explain in your own words what functional size measurement is and how it works.
Summarize the main ideas of the reading.

With both questions, students are pushed to recall (remember) and summarize (understand) what they read. However, the teacher has to reinforce and complement, in class, the main points of the reading.

It is important to emphasize that students should know in advance that the understanding of the reading will be assessed. Teachers should keep in mind that is preferable to assign only key readings. Overloading students neither let them focus on key issues nor reflect on the topics that they are expected to learn.

AT5-2: Final Exam

The questions in the final exam have to assess the understanding of the concepts learned in class. Consequently, the questions may be: an open question about concepts of measurement; and/or one exercise for obtaining the functional size of a piece of software.

AT5-2: Final Exam - Open question

Since the ILO5-1 aims to assess understanding in students, open rather than closed questions are preferable for determining the depth of the student's understanding (Biggs and Tang, 2007). Examples of open questions are:

Explain in your own words why software measurement is important and what its potential benefits are for a software organization.
Explain why and how the COSMIC method is used in the software development process.

Guidelines on how to evaluate the levels of understanding of the students based on their responses are explained in the SOLO taxonomy (Atherton, 2013; Biggs and Tang, 2007) -see chapter 1. For the above mentioned question 2, an example of a rubric (see Table 4.4) based on the SOLO taxonomy is proposed, along with the analysis of levels of understanding of four hypothetical answers provided by students.

Table 4.4 Rubric for an open question (adapted from (Biggs and Tang, 2007))

Excellent	Good	Adequate	Marginal
Able to explain" why" and "how" in a coherent way, including details and giving his/her own opinion, point of view, using his/her own words. Able to link the use of COSMIC with real-life professional contexts.	Able to explain" why" and "how" in a coherent way, including details and giving his/her own opinion, point of view, using his/her own words.	Able to explain" why" and "how" giving few details and using words and expressions provided in class.	Able to briefly write about "why" and/or "how" the method is used in the software development process.
A+, A, A-	**B+, B, B-**	**C+, C, C-**	**D**

By using the SOLO taxonomy, an analysis of possible answers provided by students to question 2 is presented next:

Answer 1: *COSMIC is a method that measures the functionality of the software. This is why this method can be used to measure the size of a piece of software based on the functional user requirements.*

Analysis of answer 1:

Level of understanding: 2 - Unistructural

Explanation: The student's answer is incomplete because it only refers to a brief definition of the COSMIC method. The student excludes the explanation of the connection of COSMIC with the software development process.

In the rubric: Marginal

Answer 2: *COSMIC is a method that measures the functionality of software without including technical or quality considerations. COSMIC was created to overcome the weaknesses of the existing Function Points methods which were not adequate to measure the functionality of real-time and embedded software. This method can be used during the*

software development process to measure the size of the software, as a whole or in parts. The size is based on the functional user requirements. The size obtained is useful for allocating resources to software projects.

Analysis of answer 2:

Level of understanding: 3 - Multistructural

This answer adds more details of COSMIC and how it is used in the software development process; however, the student does not really explain how COSMIC is related with the software development process.

In the rubric: Adequate

Answer 3: *COSMIC is used for software developers because it is a method that measures the functionality of software. The method only considers functional user requirements and not technical or quality aspects of the software. COSMIC was created to overcome the weaknesses of the existing Function Points methods which were designed to measure only business software. Any kind of software can be measured with the COSMIC method: business, real-time and embedded software.*

In COSMIC, the functional user requirements (FUR) are represented by one or more functional processes which are also represented by four data movements: entry, exit, read and write. Each data movement is equivalent to one CFP (COSMIC Function Point), which is the standard unit of measurement in COSMIC. COSMIC can be used to measure FURs in any phase of the software development process. In addition, the method is also useful to measure software at any level of decomposition, this means: as a whole or components or sub components. Finally, it can be used in any layer of multi-layer software architecture.

Analysis of answer 3:

Level of understanding: 4 - Relational

The answer explains well the relationship between COSMIC and the software development process.

In the rubric: Good

Answer 4: *COSMIC is used for software developers because it is a method that measures the functionality of software. The method only considers functional user requirements and not technical or quality aspects of the software. COSMIC was created to overcome the weaknesses of the existing Function Points methods which were designed to measure only business software. Any kind of software can be measured with the COSMIC method: business, real-time and embedded software.*

In COSMIC, the functional user requirements (FUR) are represented by one or more functional processes which are also represented by four data movements: entry, exit, read and write. Each data movement is equivalent to one CFP (COSMIC Function Point), which is the standard unit of measurement in COSMIC. COSMIC can be used to measure FURs in any phase of the software development process. In addition, the method is also useful to measure software at any level of decomposition, this means: as a whole or components or sub components; and in any layer of multi-layer software architecture.

The applicability of the method to a variety of types of software may be one of the reasons why it has been adopted worldwide by software developers. This adoption might be considered as beneficial for software developers since it is possible for them, for example, to submit their data to a repository of software projects. One of the well known repositories is ISBSG and based on the data in the repository, they can do benchmarking with respect to other projects with similar characteristics. Finally, the whole software community can benefit from the COSMIC method since the data from measured projects can be used by software measurement researchers to develop estimation models.

Analysis of answer 4:

Level of understanding: 5 - Extended abstract

This answer meets the inquiries of the question and gives the notion that the student can hypothesize (very high level of understanding) about the benefits of using COSMIC.

In the rubric: Excellent

AT5-2: Final Exam - Exercise to measure the functional size of a piece of software

Solving an exercise during the final exam allows teachers to assess individual performance in students when measuring a piece of software (understand and apply).

Regarding the measurement exercise, this may contain a small functionality. It does not have to be identical or somehow similar to previous examples presented in class. The exercise should be something simple while at the same time challenging, so that students are expected to think out a solution for the problem. If identical or similar exercises are used, the students will recall information instead of apply what they have learned in a new situation. The exercise - and assignments in general - should not be too difficult with unrealistic expectations. Table 4.5 presents an example of a rubric for grading the exercise.

Table 4.5 Rubric for functional size measurement

Complete	Partial	Minimal	Null
Able to identify all the triggering events, functional processes, data groups and more than 80% of the data movements.	Able to identify a considerable number of triggering events, functional processes, data groups (75%) and can distinguish the different types of data movements.	Able to identify some triggering events, functional process and data groups (up to 50%). Struggle identifying data movements.	Able to identify few triggering events, functional process and data groups (up to 25%). Not able to identify data movements correctly.
A+, A, A-	**B+, B, B-**	**C+, C, C-**	**D**

AT5-3: Group project

One of the assignments that help students to reach higher levels of learning is individual or group projects. The intent behind a project is to put knowledge to work in a real-life

professional problem (Biggs and Tang, 2007), which cannot be performed in a two or three hours final exam. In the case of group projects, they also help students to improve their interpersonal and persuasive skills. In this sense, it is preferable for students to work in a group with others rather than their close friends in order to develop those skills (Romiszowski, 2009). Forming groups and promoting team building among them can be performed by using proper games or techniques such as the ones included in (Network, 2006).

Regarding the project, teachers can ask students to measure the functional size of the entire small-well-documented software application that they are required to develop in the course; otherwise, students can measure only part of the software. Teachers should pay attention to the time that students spend in the project. This is important because more time does not necessarily produce more learning. Overloading students with a lot of work may stress and discourage them. Students may become tired and produce a poor project, which implies a poor learning. If a teacher considers that it would be too much work for the students, they could ask, for example, to measure only 5 key functionalities of the software. This measurement activity will encourage students to *apply* what they have learned and to reinforce their previous knowledge. Notwithstanding, a good *prompt* is always necessary to make the assignment -project- more challenging and interesting (Suskie, 2009). As an example, see Table 4.6 with the prompt designed for this project.

Aspects such as distribution of tasks and roles have to be handled carefully in a group project because students may divide themselves the tasks in such a way that they may learn little. It is advisable to ask students to do peer-assessment and to write a short self-reflection about the project.

The peer-assessment should be performed in secret and through a rubric designed for this purpose (Biggs and Tang, 2007). Examples of peer-assessment rubrics can be found in (Suskie, 2009) as well as on the Internet.

Table 4.6 Prompt for a project that includes functional size measurement

Among the learning outcomes for this course, you have to: use measurement units and scale types, follow a measurement technique, measure time and effort, and obtain the functional size of a small-well-documented set of simple functional requirements.

To help you reach these learning outcomes, your major task in this course is to work in a group project in which you will measure time, effort and functional size. For the measurement part of this project, you have to consider the following three key areas: preparation, execution, and report.

Preparation:

- Download the data collection questionnaire for *Development and Enhancement* - COSMIC DCQ - available in the website of ISBSG (The International Software Benchmarking Standards Group Limited) http://www.isbsg.org/ISBSGnew.nsf/WebPages/286528C58F55415BCA257474001C7B48?open.
- Become familiar with the seven sections of the questionnaire.
- Read the questions in each of the sections and identify which of those you could answer based on the data you will collect.
- Pay attention on how the following terms in the questionnaire are used: units, scales, measurement process, time, effort and functional size.

Execution:

- Analyze the functional requirements and identify: the functional users, the triggering events, the functional processes, the data groups and the data movements.
- Obtain the functional size of the software
- Keep a record of the time spent in the project' activities
- Calculate the total duration (time) and effort (person/hours) spent in the project
- Fill the data collection questionnaire (from ISBSG) by answering the questions for which you have the available information.

Report:

- Write an introduction of your project (no longer than one page).
- Include one or two sheets with a table and graphics showing the time and effort spent in the project.
- Include a sheet with the calculation of the functional size similar to the one used in class.
- Write a reflection based on the results obtained by the group, in terms of the functional size of the software, the total time and effort spent in the project. The reflections should mention at least what the group did well and wrong, and what can be improved. Feel free to include other aspects that you may consider important for a good reflection.
- Include an individual reflection -one page of length per group member- where each student states what his/her contribution in the project was and how this allowed his/her to reach the learning outcomes.
- Use proper and clear language to write the report.
- Look for references -if necessary-to support your conclusions.

The self-reflection is very important because it "helps students learn" (Suskie, 2009). The reflection -one page of length- should state what the student's contribution in the project was and how this allowed his/her to reach the learning outcomes (see section *Report* in the table 4.6). This individual student reflection could be either graded or not. If it is graded, students

may tend to sugarcoat what they actually learned; however, the grading will encourage students to do the task and write carefully their reflections of learning (Biggs and Tang, 2007; Suskie, 2009). Whether or not the reflection report is graded, the writing activity aims to develop in students the critical thinking skill (Garrison and Archer, 2000). Also the reflection exercise may help them improve their metacognition skill (learn how to learn) and the ability for making synthesis (Suskie, 2009). To enhance the effectiveness of critical thinking, a presentation of the project could be requested by the teacher (if the timeframe of the course allows doing so). The teacher, for example, could appoint a jury composed of two teachers (colleagues), three classmates and one software practitioner to grade the students' project presentation (Villavicencio and Abran, 2011a). Once again, writing and talking are crucial for knowledge construction (Garrison and Archer, 2000) (Hagström and Lindberg, 2012).

Finally, based on the performance achieved by students, the teacher should provide in-person feedback (individually or group) or written comments to students. The feedback should give information about the project per se and the process followed by students. Directions of how to improve them (project, process) and clarifications of concepts and procedures that were not completely understood can be considered as effective feedback for students (Hattie and Timperley, 2007). It is important to remark that feedback should not necessarily be accompanied by a grade. Indeed, studies have shown that written comments alone heighten the student's learning (Hattie and Timperley, 2007).

4.7 Achieving meaningful learning and developing skills via the framework

As seen in chapter 1, promoting meaningful learning (i.e. having relevant knowledge that can be used to solve new problems and to understand new concepts) requires the use in combination of four vital activities: listening, talking, reading and writing. In this respect, Hagström and Lindberg (2012, 122) argue that "learning and the use of language are inseparable" (Hagström and Lindberg, 2012). This means that students must write and

communicate with others their reflections, comparisons, analysis, and conclusions to demonstrate their learning growth (Hagström and Lindberg, 2012).

In this chapter and in the other four examples included in Appendix XV, the four vital activities have been exploited not only to promote meaningful learning in students but also to expand the development of communication, interpersonal (ability to communicate with others) and thinking skills (the ability to engage in reflective and independent thinking).

Accordingly, listening to a lecture contributes to create meaning; however, this activity could have a short-lived effect without the other complementary activities (in class activities, discussion, reading, group project and so on). In the case of reading, it gives good possibilities for reflection (during the time devoted for reading) which facilitates the achievement of higher levels of cognitive processing. Besides, talking helps students to identify inconsistencies as well as narrow and limited thoughts or perspectives, which also contributes the development of a critical way of thinking. Furthermore, writing is very effective for this purpose because during the writing process, students need to think reflectively in order to write in a coherent way by organizing their thoughts (Garrison and Archer, 2000).

This framework should be considered as a point of departure for teaching software measurement. We suggest personalizing this resource (the framework) according to: your teaching style, time constraints, learning environments, and the students' needs. The personalization can facilitate the reaching of the expected learning outcomes and can expand the possibilities for developing communication, interpersonal and thinking skills in students. Undoubtedly, the development of skills cannot be achieved in one single course; it should be performed throughout the whole university program. Finally, the examples presented as part of this framework should not be taken as the unique possibility to teach software measurement for undergraduates. Certainly, with your imagination, many other creative ways to teach the proposed content and to use the four vital activities may emerge from the revision and/or usage of this framework.

CHAPTER 5

EVALUATION OF THE PROPOSED FRAMEWORK

This chapter focuses on the evaluation of the proposed educational framework. The activities performed within this evaluation phase are shown in Figure 5.1 and presented in details in this chapter.

5.1 Purpose and Scope of this evaluation

The purpose of evaluating the proposed framework is to determine the extent to which the framework is perceived as useful for the enhancement of software measurement education at the undergraduate level prior its deployment in a university setting.

The scope of the evaluation is limited to analyzing the perceptions of university teachers about the impact that the framework may have in an educational context. The evaluation does not include the implementation of the framework in the academic environment. However, previous to the evaluation by teachers, the understandability of the activities and examples proposed in the framework were evaluated by novices in software measurement.

5.2 Evaluation criteria

The criteria employed for evaluating the proposed educational framework have been adapted from the theoretical model of Gopal *et al* 2002. The Gopal *et al.* model was used to test the effects of various factors that affect metrics programs success (Gopal et al., 2002). Other evaluation works (Gresse von Wangenheim, Thiry and Kochanski, 2009; Kay, 2011; Stamelos et al., 2000) - focused on learning objects, learning games, and educational software - were also used to develop a new evaluation model for the present thesis (see Figure 5.2). This new model is meant to investigate the factors (Factor 1 and Factor 2 from figure 5.2) that may influence teachers on adopting the educational framework.

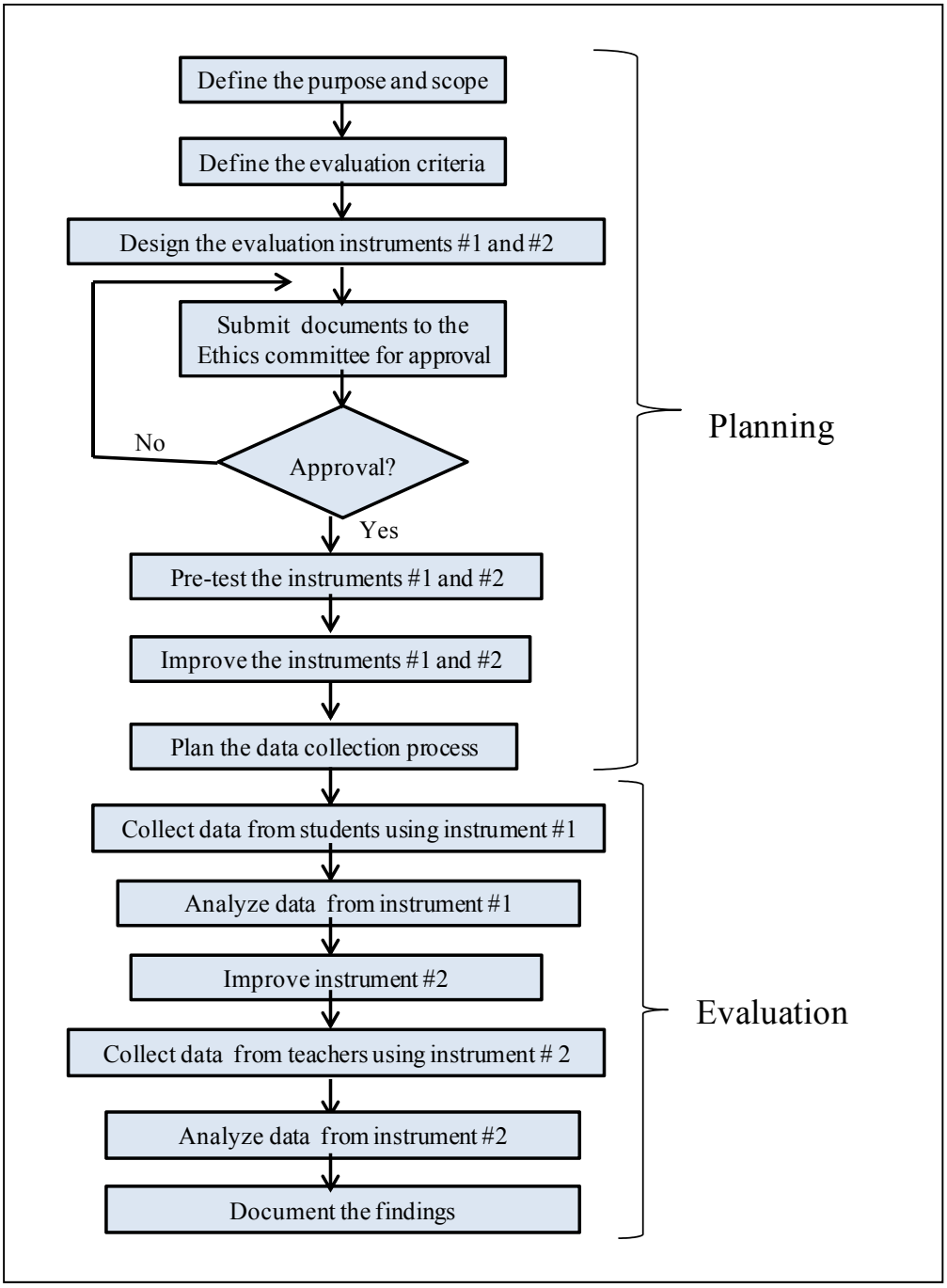

Figure 5.1 Detailed activities for evaluating the proposed educational framework

According to our model, the framework's design (Factor 1), on the one hand, is expected to have an effect on the usage and perceived usefulness of the educational framework. On the other hand, the engagement of academia -teachers and authorities- (Factor 2) can influence the perceived enhancement of the teaching and learning process in software measurement. In addition, the usefulness of the framework can positively affect the stakeholders' perception that the teaching and learning of software measurement can be improved.

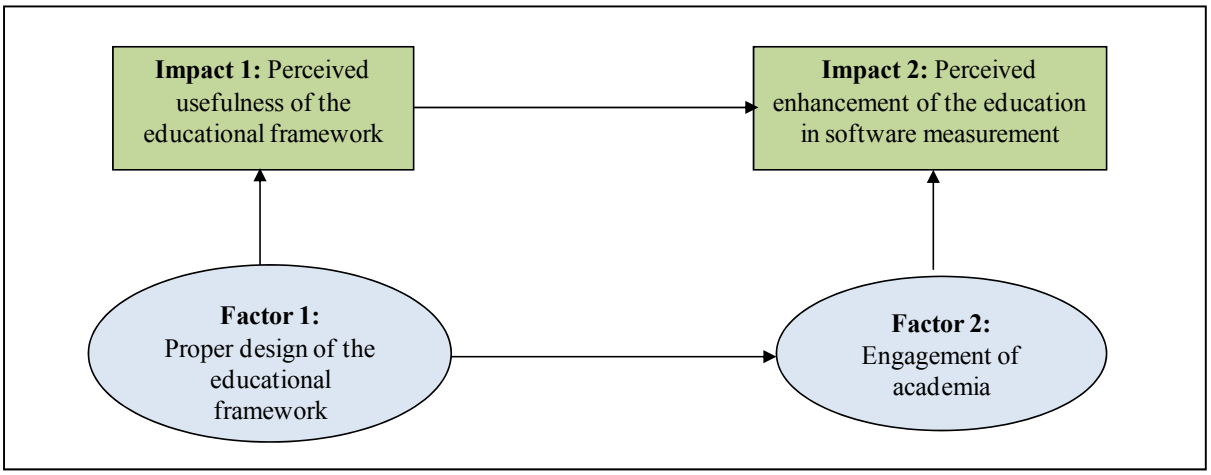

Figure 5.2 Model to evaluate the proposed educational framework, adapted from Gopal *et al.* 2002

As shown in Fig. 5.2, factors 1 and 2 may influence the adoption of the proposed framework (i.e. Impact 1: Usefulness **USEF**) and the enhancement of software measurement education at the undergraduate level (i.e. Impact 2: Enhancement **ENHA**).

In the case of factor 1, the issues involved are: the content, and the friendliness of the framework. In the case of factor 2, the issues consist of: the willingness of teachers to adopt the framework, and the facilities provided by universities for facilitating the adoption of the framework. These issues are based on the work of (Anderson et al., 2001; Biggs, 1995; Biggs and Tang, 2007; Kay, 2011; Stamelos et al., 2000). The issues per type of factor are listed next.

Factor 1: Proper design of the educational framework

Issue 1: Content of the framework (**CONT**)

- Defined ILO (Intended Learning Outcomes) for each topic to be covered.
- Availability of teaching materials (slides, exercises, bibliographic references, useful links, etc)
- Description of how to conduct teaching and learning activities (lectures, work groups, discussion, etc)
- Availability of examples regarding activities and assessment tasks
- Rubrics with criteria for assessment of learning

Issue 2: Friendliness of the framework (**FRIE**)

- Easy to use.
- Understandable
- Not boring
- Includes Tables and figures.
- Well organized

Factor 2: Engagement of academia

Issue 1: Willingness of university teachers to use the framework (**WILL**)

- Motivation of teachers on the use of the framework
- Alignment of teachers with the proposed framework (current trends of higher education practices)
- Awareness of the importance of software measurement

Issue 2: University facilities (**FACI**)

- Availability of resources and learning environments for using the framework (classrooms for group work, audiovisual aids, books, etc)
- University support (openness to adopt a constructivist approach)

Regarding the Impacts 1 and 2 shown in Figure 5.2, they represent the dependent variables of Factors 1 and 2. This means that a positive relationship exists between the framework's design (in terms of its capacity to attract teachers - **CONT** & **FRIE**) and the teachers' perceptions of its usefulness as a resource for teaching and learning. Because of its usefulness (**USEF**), teachers would be willing to adopt the framework (Factor 2: Willingness - **WILL**) which, in turn, may enhance software measurement education (Impact 2: Enhancement - **ENHA**). To accomplish this, support from universities (in terms of resources and policies - **FACI**) is needed to promote new trends in higher education.

In the evaluation model, the two factors (Factor 1 and Factor 2) and their impacts (Impact 1 and Impact 2) are constructs - an abstraction of a subject that is being studied. A construct cannot be observed and measured directly; therefore, it needs to be inferred through observable and directly measured variables (SAGE Research Methods, 2013).

The next section describes the instruments used for the evaluation of the framework, including a detailed explanation of the constructs.

5.3 Instruments

To evaluate the framework, two instruments were designed; one for learners and the other for university teachers. The ethics committee of ETS gave the approval for performing the evaluation of the framework and using the instruments on February 28th 2013.

5.3.1 Instrument designed for learners

The objective of this instrument (instrument #1) is to know the perception of students about the understandability of a sample of software measurement examples and assessment tasks (exercises, readings, project, etc) to be included in the framework.

The instrument designed for this purpose is a questionnaire containing two sections: 1) general information and 2) evaluation of the understandability of examples and tasks (see appendix XVI).

The section 1 (general information) includes three questions about the previous knowledge of software measurement, while section 2 includes five activities to inquire learners about the improvements needed to make the examples and tasks clearer and more understandable. Section 2 is meant to receive written feedback from the learners for building a useful framework. That is, a framework that contains: guidelines with clear examples of how to measure the functional size of a piece of software, and comprehensible instructions for learners of how to perform tasks proposed in the framework to assess their level of learning.

Each of the five activities included in the questionnaire requires from learners: indicating the level of agreement with statements related to the understandability of the examples or tasks; circling the words or phrases that were not understood; and providing suggestions of how to improve those examples or tasks (i.e. what to add or change). Each activity was measured through a set of statements by using a five points Likert scale - from *Strongly agree* (5) to *Strongly disagree (1).*

The five activities are:

Activity 1: Questions for an assigned reading
Activity 2: Basic example of measuring functional size (Register a new customer)
Activity 3: Example of measuring functional size where an interaction with other system is required (Withdrawal from an ATM machine).
Activity 4: Class assignment (obtain the functional size of a purchase order by working in groups)
Activity 5: Project assignment (compendium of the material learned in the course - group project)

5.3.2 Instrument designed for teachers

The objective of this instrument (instrument #2) is to determine the level of adoption of the proposed framework by its potential users. A questionnaire - in English and Spanish - was designed for this purpose.

The questionnaire is divided into two sections: 1) General information and 2) Evaluation of the proposed framework.

Section 1 includes four questions related to the teaching experience of the participants, and section 2 includes the statements for the four constructs (Factor 1, Factor 2, Impact 1 and Impact 2).

Each construct was measured through a set of statements by using a five points Likert scale - from *Strongly agree* to *Strongly disagree*. In the case of constructs 1 and 2, both were divided in two sub-constructs.

The statements used for measuring each construct are the following:

Construct 1: Factor 1 (Proper design of the educational framework)

Sub-construct 1-1: Factor 1, Issue 1 (Content of the framework - **CONT**)

- Having a suggested content for the teaching of software measurement would be a valuable resource for teachers.
- Availability of a set of suggested learning objectives would facilitate teachers to evaluate the students' performance.
- The learning objectives suggested in the educational framework are useful in guiding teachers to focus their attention on relevant software measurement topics.

- The accessibility to teaching materials related to software measurement (e.g. bibliographic references, rubrics, examples of exercises, useful links, etc.) would facilitate the teaching of software measurement.
- Having a set of suggested activities for teaching and learning software measurement (e.g. lectures, class discussion, workgroups, etc.) associated to the learning objectives would increase the odds that students accomplish the expected outcomes.
- The availability of rubrics that include assessment criteria would provide an objective guidance for assessing the students' performance in software measurement.
- The content of this framework (learning objectives, teaching activities, rubrics, etc.) is relevant to my interests in teaching software measurement.

Sub-construct 1-2: Factor 1, Issue 2 (Friendliness of the framework - **FRIE**)

- The framework is presented in a way that has kept my attention during the whole reading
- The figures and tables helped me to understand the examples provided in the framework
- The examples included in the framework are easy to follow
- The instructions for the activities and task are easy to follow
- The framework is easy to use
- The framework is well organized
- This framework is understandable
- The framework made me learn about teaching and learning in a pleasant way

Construct 2: Factor 2 (engagement of academia)

Sub-construct 2-1: Factor 2, Issue 1 (Willingness to use the framework - **WILL**)

- I will use this framework to teach my courses
- I would like to read more examples included in this framework
- I liked the material included in the framework (content, teaching and learning activities, assessment tasks, rubrics, etc.)

- I found the framework engaging
- I would recommend the proposed educational framework to my colleagues
- Having an educational framework suggesting content, teaching materials, learning objectives, assessment criteria would serve as a motivating factor for its adoption among teachers
- Availability of an educational framework would raise teachers' enthusiasm exploring new ways of teaching of software measurement topics
- Teachers would be willing to adopt an educational framework aligned with the current trends of higher education practices
- Due to the fact that the educational framework emphasizes the importance of software measurement, it would encourage teachers to its adoption.

Sub-construct 2-2: Factor 2, Issue 2 (University facilities - **FACI**)

- For the adoption of the proposed educational framework, teachers would require the support of university principals (i.e. departmental chairs) for adjusting their courses to embrace the suggested content, learning objectives, teaching methods, learning activities and assessment criteria.
- Having the support from university authorities for making changes in courses related to software measurement would facilitate teachers to adopt the proposed educational framework.
- The availability of learning environments (i.e. having the required labs, audiovisual facilities, classroom distribution for allowing group work) would facilitate the adoption of the proposed educational framework.

Construct 3: Impact 1 (Usefulness of the framework - **USEF**)

- I think that the proposed educational framework would be useful for the teaching and learning of software measurement at the undergraduate level.

- Having an educational framework would facilitate the development of a common curriculum of software measurement for undergraduate students.
- The proposed educational framework would awaken students to the relevance of software measurement in being better prepared for the software industry demands.
- I have positive feelings on the usefulness of the proposed framework.
- For learning purposes, this framework is very useful.
- The examples provided in the framework are helpful.

Construct 4: Impact 2 (Enhancement of the education - **ENHA**)

- An educational framework, commonly used by teachers, would facilitate the development of joint projects among scholars. The projects could be related to the improvement of the education in software measurement for undergraduates.
- The improvement of the teaching and learning process in the software measurement domain would be possible by using this framework.
- The proposed educational framework would contribute to the development of the students' critical thinking skills that need to be developed during their university career.
- The type of activities promoted in the educational framework would facilitate the enhancement of the students' problem solving skills.
- The teaching and learning activities suggested in the educational framework would facilitate the enhancement of students' teamwork skills.
- The proposed framework would facilitate teachers to better prepare students on the fundamentals of software measurement allowing them to initiate in activities related to software process improvement in organizations.
- By reading this framework, I have gain knowledge about general teaching and learning practices.
- As a teacher, I feel that I have learned while reading this framework.
- The application of this framework can help students achieve the learning goals.
- The framework contributes to the learning of software measurement.
- Overall, the proposed framework will help students to learn software measurement.

- In general, the framework will help teachers improve their teaching of software measurement.

It can be observed that each construct has several statements. This is necessary to test the reliability of the constructs. This means that the responses to the statements within the same construct should demonstrate correlation. A high correlation ensures that the constructs are measured properly (Gopal et al., 2002).

The statements in the constructs were arranged randomly to avoid the respondents' bias in choosing similar value for each statement's answer. This means that we wanted to diminish the respondents' tendency to agree with the statements independent of content. In addition, the respondents (evaluators) were asked to:

- work as quickly as they could without thinking too deeply about any statement and going back to review responses given to similar statements, while marking down their first thought -*Strongly agree* to *Strongly disagree* (Pre-test with paper questionnaire)
- answer the questionnaire by taking into account that the system would not allow them to go back to the previous page to check their answers - *Strongly agree* to *Strongly disagree* (Evaluation with an online questionnaire)

The questionnaire is included in Appendix XVII.

5.4 Pre-test of the instruments

5.4.1 Pre-test of the instrument #1 - for learners

The pre-test of this instrument consisted of the revision of the wording of the questions and the organization of the activities contained into the questionnaire. The pre-test was conducted with two PhD students and two teachers.

Few changes were suggested by the participants, as follows:

- Do not present two activities in the same page; use a new page to start an activity.
- Be consistent with the use of words; avoid using different names for the same subject/object.
- Instead of using the word "task", specify the type of task (reading, exercise in class, project) from which an answer/opinion from students is expected.
- Instead of using the word learning, be more specific about what the students should be able to do (e.g. measure the functionality of the software).

5.4.2 Pre-test of the instrument #2 - for teachers

The pre-test of this instrument was performed in Lima, Peru during the Ibero-American software engineering conferences. Nine participants (university teachers) attending the four-hour tutorial named "A Constructivist Approach for the Teaching of Software Measurement" received training on the concepts and application of the proposed framework. The training also included some of the examples presented in chapter 4 of this thesis.

The instrument was also tested by three other professors who were not participating in the conferences. Two of them work on research in higher education, and the third has special interest in teaching functional size measurement for undergraduates.

From the pre-test, the following changes were suggested for the framework and the questionnaire:

- Avoid using acronyms (TLA, AT, ILO), especially at the beginning of the examples.
- Define the approximate time required for each activity performed in class.
- Rearrange the order of the material presented in the framework.
- Include guiding reading questions.
- Give step by step guidelines for group work activities.

- Explain how the feedback has to be implemented.
- Include self and peer assessment.
- Change the wordiness of some questions in the questionnaire.

The suggestions were taken into consideration for the development of the complete framework (chapter 4 and appendices XV and XXVII) and the final version of the evaluation questionnaire addressed for teachers.

5.5 Data collection and findings from instrument #1 (learners)

5.5.1 Data collection from potential learners

The data collection was performed at the beginning of June 2013 with 12 voluntary participants: 4 students who were taking the course of Software Measurement at ETS, 1 foreign student doing an internship at ETS, 3 PhD students, 3 practitioners with a bachelor diploma in computer science working on software related areas, and 1 practitioner from the telecommunication sector. Eight participants did not have previous knowledge of software measurement and four had little knowledge acquired in courses such as: project management, software engineering and software quality.

The participants took 30 to 70 minutes to review the examples and answer the questionnaire, plus a short break of 5 minutes. The purpose of the break was to keep them attentive to identify improvements needed in the educational material included in the questionnaire (functional size measurement examples and assessment tasks). When the participants finished answering the questionnaire, I reviewed their responses with each of them to assure that I understood all their comments and suggestions. Special attention was given to the participant from the telecommunication sector since we wanted to test the pre-requisites (see chapter 4) needed to learn from the examples proposed in the framework.

5.5.2 Findings

The tables 5.1 and 5.2 show the results obtained for each activity. The second column of the table represents the statistical mean of the level of agreement expressed by participants (inexperienced students and practitioners) regarding the statements formulated for each of the five activities (see section 5.3.1). The results in terms of level of agreement show that the activities #1 and #4 are less clear and understandable than the other three activities. Therefore, both activities are less likely to contribute with the expected learning outcomes and need to be improved for understandability and clearness purposes. The suggested improvements are shown in the third column of the table. All of them were made to develop the whole version of the framework.

Collecting data from inexperienced students and practitioners was good in the sense that it allowed the identification of strengths and weaknesses in the sample of examples used for testing the understandability of the educational material. Such strengths and weakness are summarized as follows:

Strengths:

- Interface: The use of graphics/figures/tables to explain the two measurement examples are useful to understand the functionalities to be measured.
- Detailed guidelines for the group project: The detailed instructions provided to students to perform the project are practical. With them, students know what to do and what the teacher expects from them.

Weaknesses:

- Ambiguity of questions for Activity 1 (guiding questions for a reading): The potential learners suggested rewriting questions 1 and 3 because they were not clear enough and several interpretations were possible to infer.
- The lack of written specifications for Activity 4 (exercise in class): The potential learners consider that the explanations of the teacher may not be enough for performing the

exercise in 20 minutes. So, the learners suggested including the written specifications, similar to what was performed in activities 2 and 3.

- Highlight learning outcomes: The learners suggested highlighting the learning outcomes because, at the end, they did not remember what the learning purpose was.

All of these suggested changes were used to enhance the understandability and readability of the examples and assessment tasks included in the framework to be evaluated by university teachers. Examples of the improvements made in the framework are:

- The addition of footnotes in the tables that show the calculation of the functional size of the software reminding the meaning of E, X, R and W.;
- The reduction of complexity of Activity 4 (exercise in class to practice functional size measurement) in order to assure that the exercise can be performed in 20 minutes.
- The inclusion of written specifications in the Activity 4 to complement the explanation provided by the teacher.

5.6 Data collection and findings from instrument #2 (university teachers)

5.6.1 Data collection

At the end of August 2013, we sent invitations by e-mail to 34 university teachers who participated in the previous studies and manifested their interest in knowing and evaluating the proposed educational framework. The invitation contained a link to the questionnaire developed with LimeSurvey 2.0 which appears in the appendix XVII. From the date that teachers received the invitation, they had 30 days to review the framework and return the evaluation questionnaire (instrument #2). Since only few teachers met with the deadline, it was necessary to extend the evaluation period until mid of November. During this period, only 21 teachers returned the evaluation questionnaire. As it was difficult to find more respondents, the data analysis was performed with this number of responses.

Table 5.1 Findings of the evaluation performed with instrument #1(learners)

Activity 1: Questions for an assigned reading		
Statements	**Mean**	**Improvements needed**
Guiding question #1 is clear/understandable	4	1. Split question #1 in two questions. 2. Rewrite question #3 to make it unambiguous.
Guiding question #2 is clear/understandable	5	
Guiding question #3 is clear/understandable	4.5	
The guiding questions are helpful for students	4	
Activity 2: Basic example of measuring functional size (Register a new customer)		
Statements	**Mean**	**Improvements needed**
The example is clear (the functionality to be measured is understandable)	5	1. Explain the acronyms (E, X, R, W, CFP). 2. Add an extra explanation about the counting of data movements when the user edits the customer data. 3. Give more emphasis to the arrows that show the data movements
The procedure followed in the example is easy to understand	4.5	
The figure helps you to understand the example	5	
After reading the example, you can measure a small-well-documented set of simple functional requirements	5	
Activity 3: Example of measuring functional size where an interaction with other system is required (Withdrawal from an ATM machine).		
Statements	**Mean**	**Improvements needed**
The example is clear (the functionality to be measured is understandable)	5	1. Add an explanation for each data movement, especially when there is a communication with other systems. 2.Write all the assumptions (pre-conditions) to perform the withdrawal functionality in order to make the example clearer. 3. Include the steps that have to be followed for measuring the Functional Size.
The procedure followed in the example is easy to understand	4.5	
The figure helps you to understand the example	5	
After reading the example, you can measure a small-well-documented set of simple functional requirements	4	

Table 5.2 Findings of the evaluation performed with instrument #1(learners)

Activity 4: Class assignment (obtain the functional size of a purchase order by working in groups)		
Statements	**Mean**	**Improvements needed**
The user interface is easy to understand	4.5	1. Add written specifications to complement the explanation of the teacher. 2. The written specifications are necessary to understand the data model and user interface without written specification. 3.The user interface should show the sequence of steps to enter a purchase order. 4.Allocate more time to make this exercise in class.
The data model is easy to understand	4.5	
The instructions provided by your teacher are clear (You understand the task that you have to do)	3.5	
The time assigned to perform the task is reasonable	3	
Working with classmates allows to gain and to share knowledge	4.5	
Working with classmates will facilitate the development of this exercise	4.5	
By performing this exercise you will improve your knowledge of functional size measurement	4	
Activity 5: Project assignment (compendium of the material learned in the course - group project)		
Statements	**Mean**	**Improvements needed**
The instructions to perform the project are clear (You understand what you have to do)	5	1.The ILOs (intended learning outcomes) should be highlighted in the explanation of the project. 2.Specify the difference between time and effort 3.Add the link of the ISBSG website in the instructions
Working with classmates allows to gain and to share knowledge while developing the project	4.5	
Working with classmates will be helpful to perform the project (sharing tasks, identifying mistakes, having more ideas to do a better job)	4.5	
By performing the project you will be able to reach the learning outcomes	4.5	

5.6.2 Findings

From the 21 participants:

- 57% has more than 10 years of teaching experience in topics related to software measurement;
- 43% works in software engineering departments; and 52% in computer science departments at universities;
- 90% teaches courses related to software measurement for both undergraduate and graduate students.

Table 5.3 presents the means (M), standard deviations (SD), Cronbach's alpha coefficients and intercorrelations among the variables of interest for the evaluation of the framework; these are: CONT, FRIE, WILL, FACI, USEF, ENHA (see section 5.3.2). The numbers in bold represent the Cronbach's alpha coefficients which denote the internal consistency between the items (questions) of the instrument used in the evaluation (questionnaire). The higher the coefficient, the higher the reliability among the questions of the same construct (Webster, 2000). Generally, acceptable coefficients for survey research should be higher than 0.7 (Gopal et al., 2002). All the coefficients were well above this cut-off point, except for one of them that is close to this value, which is acceptable for a newly developed measure (Nunnally and Bernstein, 1994).

Table 5.3 Correlation matrix

	M	SD	1-1	1-2	2-1	2-2	3	4
1-1 CONT	4.456	0.416	**0.788**					
1-2 FRIE	4.405	0.559	.615**	**0.867**				
2-1 WILL	4.280	0.471	.811**	.666**	**0.801**			
2-2 FACI	3.683	0.734	0.342	0.207	0.174	**0.619**		
3 USEF	4.500	0.435	.831**	.755**	.724**	0.287	**0.708**	
4 ENHA	4.321	0.433	.795**	.664**	.716**	0.337	.867**	**0.882**

N=21; ** The correlation is significant at p < .01 (bilateral)

In Table 5.3, the numbers below the diagonal represent the correlation value between each pair of variables. From this table, it can be observed that all variables except FACI (subconstruct 2-2) are correlated at the 0.01 significance level. This result indicates that the variable FACI does not have a relationship with the other variables. As described in section 5.3.2, FACI is a sub construct involving two factors: one is the availability of resources and learning environments for using the framework; and the other refers to the university support to adopt a constructivist approach. According to the results presented above neither the two predictor factors have an impact on the two dependent variables; that is, the perception on the usefulness of the framework and the enhancement of education in software measurement.

In order to test the relationship between the independent variables (content - CONT, friendliness FRIE, willingness WILL, and facilities FACI) and the dependent variables (usefulness USEF, enhancement ENHA, and willingness WILL), a regression analysis was performed (see section 5.2 for the explanation about the independent and dependent variables). Table 5.3 shows the coefficients, standard errors and level of significance of the linear relationship between the independent and dependent variables.

Table 5.4 shows the relationship between *usefulness* -USEF (dependent) and *content* CONT (independent) and *friendliness* FRIE (independent). The variable CONT is significant at 0.0001 and FRIE at 0.05. These findings indicate that both variables, CONT and FRIE, have an influence in the teachers' perception regarding the usefulness (USEF) of the framework, especially the *content*.

Table 5.4 Regression results - Dependent variable: USEF

Variable	Coefficient	Std Error	Significance p value
Constant	0.411	0.517	0.437
CONT	0.616	0.145	0.000
FRIE	0.305	0.108	0.011

F value=32.937; $p<0.0001$
$R^2=0.762$;
N=21

Table 5.5 presents the relationship between *willingness* - WILL (dependent) and *content* CONT (independent) and *friendliness* FRIE (independent). In this case, only the variable CONT is significant at 0.001. This indicates that the willingness of teachers to adopt the framework (WILL) relies on its content (CONT) rather than its friendliness (FRIE). This is a very important finding because it seems to indicate that the content is more essential than friendliness despite the effort spent in developing friendly educational resources. Notwithstanding, further research is needed to corroborate this result.

Table 5.5 Regression results - Dependent variable: WILL

Variable	Coefficient	Std Error	Significance p value
Constant	0.024	0.660	0.971
CONT	0.731	0.185	0.001
FRIE	0.227	0.137	0.117

F value=21.226; $p<0.0001$
$R^2=0.669$
N=21

Table 5.6 shows two relationships in which the enhancement of the education in software measurement (ENHA) acts as a dependent variable. The first relationship identifies if ENHA depends on: the willingness of teachers for adopting the proposed framework (WILL), and the facilities and support of university authorities for using the framework. In this regard, only the willingness of teachers (WILL) is strongly significant at 0.0001 and impacts the perception about the enhancement of the education in software measurement. The second relationship shows that the enhancement of the education in software measurement (ENHA) depends on the usefulness of the framework (USEF). This relationship is certainly significant at the 0.0001 level. Consequently, a perceived usefulness of the proposed framework by teachers leads to a perceived enhancement of the education in software measurement.

Table 5.6 Regression results - Dependent variable: ENHA

Variable	Coefficient	Std Error	Significance p value
Constant	1.175	0.663	0.930
WILL	6.240	0.146	0.000
FACI	0.129	0.940	0.185

F value=11.423; $p<0.001$
R^2=0.510
N=21

Variable	Coefficient	Std Error	Significance p value
Constant	0.434	0.516	0.411
USEF	0.864	0.114	0.000

F value=57.360; $p<0.0001$
R^2=0.738
N=21

Finally, the results of the present study should be considered with caution since the number of data is small. Therefore, generalizations should not be made from these results as more data are needed to perform more precise analysis about the relationships among the variables. As mentioned in chapter 6, future research should consider the evaluation of the framework to determine the impacts in terms of students' learning. For this kind of evaluation, a model similar to the one presented in this chapter can be used.

CHAPTER 6

CONTRIBUTIONS AND FUTURE WORK

6.1 Contributions

The research focus of this thesis work was the development of an educational framework intended to promote the achievement of learning outcomes in undergraduate students. For its development, it was necessary to determine first the state of the art of software measurement in higher education and to identify the priorities of this subject matter for undergraduates. The research findings have led to 8 publications: 6 for conferences, one accepted journal and one submitted journal - see appendices XVIII to XXV. These publications, the framework, and other documents included as appendices have been created for the benefits of the following audiences:

University teachers or instructors: The proposed educational framework provides guidelines to assist university teachers in the teaching process of software measurement topics at the undergraduate level. These guidelines follow a constructivist approach by offering a variety of pedagogical techniques through examples related to: teaching and learning activities, approaches to assess learning, and exercises - See chapter 4 and appendices XV, XX, XXV, and XXVII.

Undergraduate students attending software engineering programs or beginners in the software measurement field: The framework allows students to learn software measurement starting from the basis (Basic concepts in software measurement) and going through the measurement process, measurement techniques and specific measurement methods. In addition, students can attain the learning goals by being exposed to real life examples (e.g. measuring the functional size of a "purchase order" or "online shopping") - See chapter 4 and appendices XV and XXVII.

Related bodies of knowledge: During the research phases, several opportunities for improvement were identified in existing documents related to software measurement. In this regard, the bodies of knowledge can use the recent work as a way of enhancing the corresponding documentation - See appendix XXVIII. Also, a mapping of software measurement topics was developed among three bodies of knowledge (see chapter 1 - Table 1.2).

Software measurement organizations: These software measurement organizations can use the educational framework for providing training to practitioners. Also, they can take ideas to develop assessment activities for certification purposes (chapter 4 and appendix XV). In the case of the Common Software Measurement International Consortium (COSMIC), they have access to a new case study developed as one of the examples of the framework (see the *online shopping* example - appendix. XXVII). In addition, they have been provided with suggestions to improve the understandability of the current and upcoming versions of the COSMIC method (appendix XXVIII).

Software organizations: The educational framework can also be used for training purposes at software organizations interested in starting or improving software measurement programs (chapter 4 and appendices XV and XXVII). Moreover, some documents produced during this research are useful for benchmarking purposes - in terms of certifications, software measurement programs, tools used, knowledge of employees, software measurement priorities for organizations (see chapter 3 and appendices XXI and XXII).

Researchers in education: This doctoral work is useful not only for further research in software measurement education, but for any science or engineering field since it (this research work) provides a roadmap to: identify priorities in any field; and to develop a framework to fulfill the needs of education in a specific area of knowledge in addition to examples of how to apply software engineering educational research methodologies (chapters 3 to 5; and appendices VI to XXIX).

6.2 Implications for future research

From the findings of the literature survey, this thesis is the first research work that:

- explores software measurement in higher education in depth;
- tackles its teaching for undergraduate students by identifying the priorities of this subject matter; and
- proposes an educational framework based on the constructivist approach.

Therefore, this research can be considered as a starting point to conduct further research projects in the software measurement educational field, such as:

A. **The development a common curriculum of software measurement for undergraduate students**: This curriculum can be designed by taking as an input the proposed framework: that is, including the priorities for undergraduates (current framework) and adding complementary topics that can be covered in undergraduate courses or seminars related to software measurement.

B. **The update of the software engineering curriculum guidelines for undergraduate students in topics related to software measurement**: The curriculum guidelines can be updated by applying the suggestions presented in appendix XVIII, especially the summary of topics to be covered, levels of learning, and suggested number of hours presented in section 1, table 1 of this appendix XVIII.

C. **The enhancement of the proposed educational framework for undergraduates or beginners in the field**: Some of the foreseen opportunities of improvements are:

- Increase the type and number of teaching and learning activities and assessment tasks.
- Enlarge the number of exercises and examples for each topic, which have to be both simple and real-world situations.

- Update the current examples of functional size measurement developed with the COSMIC method version 3 to: the upcoming version 4 when this becomes available, and other functional size measurement methods like IFPUG.
- Improve the current version of the website for teaching software measurement to make it more attractive and interactive (e.g. forums, games).
- Videotape lectures and interactive lectures to make them available on the website.
- Incorporate theory and activities specifically devoted to the development of skills (i.e. for teachers who have more hours assigned to the subject matter or who are interested in developing skills in their students).

D. The creation of a research group or community focused on software measurement for higher education: A research group can be created with the university teachers who have already manifested interest in using the framework (see appendix XXX). By having a research group, the following activities can be performed**:**

1) Test and enhance the framework (iterative process):

- Test the propose framework through experimental studies with undergraduate students (*in progress*).
- Evaluate the usefulness of the framework by using a model similar to the one presented in chapter 5. The model should be tailored to determine the engagement of the students (see figure 5.2 - Factor 2) and to determine the impacts in terms of students' learning.
- Improve the framework based on the results of the evaluation.
- Conduct further experimental studies in several universities by using an enhanced, extended and standardized version of the framework.
- Compare learning results and identify new improvements needed.

2) Extend the scope of the framework (from beginners to intermediate and advance levels) by taking as an input the layer representation of the software measurement topics presented in the Figure 3.2 (chapter 3).

- Identify priorities for the intermediate and advance levels (e.g. master programs, practitioners)
- Identify levels of learning expected to achieve by students and the complementary skills
- Develop an extended version of the framework with the 2 new levels.
- Propose a common curriculum for the intermediate and advance levels.

3) Enrich the knowledge of software measurement

- Produce publications related to the education of software measurement.
- Produce a textbook of software measurements

6.3 Research impact

As mentioned before, this thesis is the first - since METKIT 20 years ago- focusing in the enhancement of the education of software measurement in universities at the undergraduate level. By continuing this line of research, a number of impacts can be foreseen in the short and long term for the academia and industry.

In the short term:

- For teachers already teaching software measurement: to have the possibility to improve their teaching of software measurement through the use of the framework and available publications which are included in http://software-measurement-education.espol.edu.ec/.
- For undergraduate students exposed to the framework: to learn software measurement in a practical way and with real-world examples.
- For new teachers and learners: raise their interest in learning software measurement, a topic rarely explored in the educational context.
- For novice professionals: access to examples and theory (easy to read) to learn the basis of software measurement and functional size measurement.

In the long term:

- At universities:
    - Breaking perceptions that software measurement is difficult and hard to learn.
    - Promoting a software measurement culture among software engineering students.
    - Adopting common software measurement terminology and practices by applying a standardized framework of software measurement or a common curriculum.
- At software development organizations:
    - With employees who are aware of the existence and importance of software measurement.
    - With employees able to participate in software measurement programs
    - With employees having knowledge to start measuring the functional size of software applications with the COSMIC method or able to learn in depth a different method (i.e. by using the basis acquire at the university).

6.4 Limitations of this research

Some limitations and validity threats of this research have been identified and summarized in the articles presented/submitted to conferences and journals (see appendices XIX, XXII and XXIII). Additional limitations are mentioned next:

- This research work mainly considers the application of the Bloom's taxonomy and partial use of the SOLO taxonomy. Others taxonomies can be used; however, this research wanted to be aligned with the current curriculum guidelines on software engineering.
- The scope of the educational framework is limited to beginners in software measurement. It covers only five software measurement topics and the corresponding levels of learning. More research is needed to extent the scope to the intermediate and advance levels.
- In general, there are only a couple of examples of suggested teaching activities and assessment tasks for each of the five topics included in the framework. More examples

are needed to better explain the suggested activities and tasks. In addition, other kind of activities and tasks can be incorporated.

- All the examples of functional size measurement were developed by using the COSMIC method v3.0.1, which was the latest published version available at the time the framework was proposed. The same examples can be developed with other methods or the upcoming new version of COSMIC.
- The framework contributes to the development of skills in an indirect way but this is not its main purpose. Complementary research is needed to look for pathways to develop skills in undergraduate students.
- Only the set of examples developed for the topic *measures for the requirements phase* has been used in a software measurement course taught at the Software Engineering Department of Izmir University of Economics (mid December 2013). This part of the framework as well as the rest of examples are expected to be used at universities in Spain in February 2014, in Ecuador in the second semester of 2014, in Peru in March 2014, and Canada.
- The proposed framework was evaluated by 21 university teachers with experience in software measurement. Despite the small number of teachers, it was possible to test the model through correlations and linear regression. The reliability test (Cronbach's alpha) demonstrates that the instrument used for the evaluation is valid and that the theoretical constructs are properly measured. Notwithstanding, another way to test the model is through factor analysis, which it could not be performed due to the limited number of evaluators. This number is limited by the fact that software measurement is a specialized field and there are not many university teachers with the expertise, time and willingness to perform the evaluation.
- The use of the framework demands: a commitment from teachers to adopt the constructivist philosophy; a strong desire to teach the course in order to reach deep learning in students; and passion and creativity to tailor the framework to serve students needs.

ANNEX I

LIST OF APPENDICES

The following list contains the appendices referenced within this thesis, which are included in the CD-ROM.

Appendix	Filename	Content
I	Appendix I Approbation finale Web survey 2011	CÉR approval of Web survey
II	Appendix II Approbation finale Delphi 2012	CÉR approval of Delphi study
III	Appendix III Approbation finale Interviews	CÉR approval of interviews
IV	Appendix IV Approbation finale evaluation framework	CÉR approval of the evaluation of the framework
V	Appendix V Approbation finale demande de renouvellement	CÉR approval of the renewal of the research project (mandatory when the project lasts more than one year)
VI	Appendix VI Questionnaires for pilot test - current state of software measurement in higher education	Questionnaires of pilot test
VII	Appendix VII Invitation Web survey by mail	Invitation web survey
VIII	Appendix VIII Questionnaire Web survey for Teachers	Questionnaire web survey teachers
IX	Appendix IX Questionnaire Web survey for practitioners	Questionnaire web survey practitioners
X	Appendix X Questionnaires Delphi Round 1	Questionnaires Delphi round No.1

Appendix	Filename	Content
XI	Appendix XI Questionnaire Delphi Round 2	Questionnaires Delphi round No.2
XII	Appendix XII Questionnaires Delphi Round 3	Questionnaires Delphi round No.3
XIII	Appendix XIII Questionnaires Verification Delphi	Questionnaires Delphi verification
XIV	Appendix XIV Questionnaires Interview Teachers	Questionnaire Interview teachers
XV	Appendix XV Set of Examples for Teaching Software Measurement	More examples of the application of the framework
XVI	Appendix XVI Questionnaire Evaluation Students	Questionnaire for the evaluation of the understandability of examples and tasks included in the framework
XVII	Appendix XVII Questionnaire Evaluation Teachers	Questionnaire for the evaluation of the framework
XVIII	Appendix XVIII - Metrikon 2010	Software Measurement in Software Engineering Education: A Comparative Analysis
XIX	Appendix XIX - JSEA 2011	Facts and Perceptions Regarding Software Measurement in Education and in Practice: Preliminary Results

Appendix	Filename	Content
XX	Appendix XX - IWSM-Mensura 2011	Educational Issues in the Teaching of Software Measurement in Software Engineering Undergraduate Programs
XXI	Appendix XXI - CCECE 2012	The Necessary Software Measurement Knowledge from the Practitioners' Point of View
XXII	Appendix XXII - submitted to journal IJSEKE	Software Measurement in Higher Education
XXIII	Appendix XXIII - SEAA 2012	Software Measurement in Software Engineering Education: A Delphi Study to Develop a List of Teaching Topics and Related Levels of Learning
XXIV	Appendix XXIV - JIISIC 2012	A Constructivist Approach for the Teaching of Software Measurement
XXV	Appendix XXV - IWSM-Mensura 2013	Towards the Development of a Framework for Education in Software Measurement
XXVI	Appendix XXVI - Educational Framework v.1	Proposed Educational Framework v.1
XXVII	Appendix XXVII - COSMIC case study	Functional Size Measurement for an Online shopping application using COSMIC

Appendix	Filename	Content
XXVIII	Appendix XXVIII - Improvements suggested to Bodies of Knowledge and COSMIC MPC	Summary of suggestions for improvement sent to Bodies of knowledge and COSMIC.
XXIX	Appendix XXIX - Results Delphi study	Summary of the results sent to the participants of the Delphi study.
XXX	Appendix XXX - Framework interest	E-mails received from teachers

LIST OF REFERENCES

Abran, A., P. Bourque and R. Dupuis. 2004. *SWEBOK: Guide to the software engineering Body of Knowledge.* Los Alamitos, California.: IEEE Computer Society.

Abran, Alain. 2010. *Software metrics and software metrology*. New Jersey: IEEE Computer Society / Wiley Partnership.

Abran, Alain 2011. *Software sizing and Cosmic ISO19761 - Power Point Presentation*.

Abran, Alain, Alain April and Luigi Buglione. 2010. « Software Measurement Body of Knowledge ». *Encyclopedia of Software Engineering*. Vol. 1:1, n° 1, p. 1157 — 1168. Consulté le 26 January 2011.

Amos, T., and N. Pearse. 2008. « The delphi technique and educating entrepreneurs for the future ». In *7th European Conference on Research Methodology for Business and Management Studies*. p. 17-24. Academic Publishing Limited, Reading, UK 2008

Anderson, Lorin, David Krathwohl, Peter Airasian, Kathleen Cruikshank, Richard Mayer, Paul Pintrich, James Raths and Merlin Wittrock. 2001. *A taxonomy for learning, teaching and assessing. A revision of Bloom's taxonomy of Educational Objectives*. New York: Addison Wesley Longman, Inc, 302 p.

Atherton, J. S. 2013. « Learning and Teaching; SOLO taxonomy [On-line: UK] ». < http://www.learningandteaching.info/learning/solo.htm >. Consulté le 25 March 2013.

Beard, Colin M., and John Peter Wilson. 2006. *Experiential Learning : A Handbook of Best Practice for Educators and Trainers*, Second edition. London: Kogan Page, 314 p.

Bhamani Kajornboon, Annabel 2005. « Using interviews as research instruments ». p. 10. < www.culi.chula.ac.th/e-journal/bod/annabel.pdf >.

Biggs, John. 1995. « Assessing for learning : some dimensions underlying new approaches to educational assessment ». *Alberta journal of educational research,* vol. 41, n° 1, p. 1-17.

Biggs, John, and Caherine Tang. 2007. *Teaching for quality learning at university*, Third edition. Buckingham: Society for Research into Higher Education & Open University Press, 335 p.

Bourque, P., A. Abran, J. Garbajosa, G. Keeni and B. Shen. 2008. « SWEBOK Version 3 ». p. 18.< http://www2.computer.org/cms/Computer.org/SWEBOK/MeasurementKA-Draft-Feb2008.pdf >. Consulté le October 15, 2010.

Bourque, P., R. Dupuis, A. Abran, J. W. Moore, L. Tripp and S. Wolff. 2002. « Fundamental principles of software engineering - a journey ». *Journal of Systems and Software,* vol. 62, p. 59-70.

Bourque, Pierre. 2013. «SWEBOK V3 Review». <http://computer.centraldesktop.com/swebokv3review/>. Consulté le September 23, 2013.

Brooks, Jacqueline G, and Martin G Brooks. 2001. *In Search of Understanding: The Case for Constructivist Classrooms*. Coll. « Merrill Education/ASCD College Textbooks Series ». Prentice-Hall, 136 p.

Buglione, Luigi. 2009. « Play'n'Learn: A Continuous KM Improvement Approach using FSM Methods ». In *4th International Software Measurement & Analysis (ISMA4)*. (Chicago, Illinois).

Bundschuh, Manfred, and Carol Dekkers. 2008. *The IT Measurement Compendium*. Springer-Verlag, 643 p.

Bush, M., and N. Ashley. 1993. « Metkit: toolkit for metrics education ». *Software, IEEE,* vol. 10, n° 6, p. 52-54.

Bush, M., and M. Russell. 1992. « Need for well-balanced education in software engineering measurement ». *Software Quality Journal,* vol. 1, n° 2, p. 81-100.

Chudnovsky, D., A. López and S. Melitsko. 2001. « El sector de software y servicios informáticos en la Argentina. Situación actual y perspectivas de desarrollo ». *CENIT, Documento de Trabajo Nº 27*. < http://www.google.com.ec/search?sourceid=navclient&ie=UTF-8&rlz=1T4ADRA_enEC341EC341&q=El+sector+de+software+y+servicios+inform%c3%a1ticos+en+la+Argentina.+Situaci%c3%b3n+actual+y+perspectivas+de+desarrollo >.

COSMIC. 2009. « The COSMIC Functional Size Measurement Method Version 3.0.1, Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761:2003) ». *The Common Software Measurement International Consortium (COSMIC)*. < http://www.cosmicon.com/portal/public/COSMIC%20Method%20v3.0.1%20Measurement%20Manual.pdf >. Consulté le 22 December 2013.

Cuadrado-Gallego, J. J., P. Rodriguez-Soria, A. Lucendo, R. Neumann, R. Dumke and A. Schmietendorf. 2012. « COSMIC Measurements Dispersion ». In *Joint Conference of the 22nd International Workshop on.Software Measurement and the Seventh International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2012*. (Assisi, 17-19 October 2012), p. 85-88.

Cuadrado-Gallego, Juan, Herrera Borja Martin and Pablo Rodriguez Soria. 2011. « Visual learning techniques for software measurement ». In *4th International Conference on Computer Science and Software Engineering*. (Montreal, Quebec, Canada), p. 59-62. 1992904: ACM.

Examiner. 2012. « Six Creative Ways to Form Groups ». < http://www.examiner.com/article/six-creative-ways-to-form-groups >.

Fetcke, Thomas. 1999. « The Warehouse Software Portfolio, A Case Study in Functional Size Measurement ». < http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.195.5828 >.

Fosnot, Catherine. 2005. *Constructivism: theory, perspectives, and practice*. New York: Teachers College Press, 308 p.

Garrison, Randy, and Walter Archer. 2000. *A transactional perspective on teaching and learning* (September 1, 2000), First edition. Emerald Group Publishing Limited, 228 p.

Gatchell, D. W., R. A. Linsenmeier and T. R. Harris. 2004. « Determination of the core undergraduate BME curriculum - the 1st step in a Delphi study ». In *Conference Proceedings. 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 1-5 Sept. 2004*. (Piscataway, NJ, USA) Vol. Vol.7, p. 5200-1. IEEE.

Gopal, A., M. S. Krishnan, T. Mukhopadhyay and D. R. Goldenson. 2002. « Measurement programs in software development: determinants of success ». *IEEE Transactions on Software Engineering, ,* vol. 28, n° 9, p. 863-875.

Gresse von Wangenheim, Christiane, Marcello Thiry and Djone Kochanski. 2009. « Empirical evaluation of an educational game on software measurement ». *Empirical Software Engineering,* vol. 14, n° 4, p. 418-452.

Hagström, Eva, and Owe Lindberg. 2012. « Three theses on teaching and learning in higher education ». *Teaching in Higher Education,* vol. 18, n° 2, p. 119-128.

Hattie, John, and Helen Timperley. 2007. « The Power of Feedback ». *Review of Educational Research,* vol. 77, n° 1, p. 81-112.

Howze, Philip., and Connie. Dalrymple. 2004. « Consensus without all the meetings: using the Delphi method to determine course content for library instruction ». *Reference Services Review,* vol. 32, n° 2, p. 174-84.

Hung, Hsin-Ling, James W. Altschuld and Yi-Fang Lee. 2008. « Methodological and conceptual issues confronting a cross-country Delphi study of educational program evaluation ». *Evaluation and Program Planning,* vol. 31, n° 2, p. 191-198.

IEEE ACM. 2004. « Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering ». Vol. SE2004 Volume – 8/23/2004, p. 135. < http://sites.computer.org/ccse/SE2004Volume.pdf >.

Integrated Software & Systems Engineering Curriculum (iSSEc) project. 2009a. « Comparisons of GSwE2009 to Current Master's Programs in Software Engineering ». *Stevens Institute of Technology*. Version 1.0. p. 96. < http://www.gswe2009.org/fileadmin/files/GSwE2009_Curriculum_Docs/ComparisonstoGSwE2009_v1.0.pdf >.

Integrated Software & Systems Engineering Curriculum (iSSEc) Project. 2009b. « Graduate Software Engineering 2009 (GSwE2009), Curriculum Guidelines for Graduate Degree Programs in Software Engineering ». *Stevens Institute of Technology*. Version 1.0. p. 124. < http://www.gswe2009.org/fileadmin/files/GSwE2009_Curriculum_Docs/GSwE2009_version_1.0.pdf >.

ISO/IEC. 2007. *Information Technology - Software Measurement - Functional Size Measurement. Part 1: Definition of Concepts. ISO/IEC 14143-1:2007* Switzerland: International Organization for Standardization, 6 p.

Iversen, Jakob, and Ojelanki Ngwenyama. 2006. « Problems in measuring effectiveness in software process improvement: A longitudinal study of organizational change at Danske Data ». *International Journal of Information Management,* vol. 26, n° 1, p. 30-43.

Izquierdo, E. 2008. « Impact Assessment of an Educational Intervention based on the Constructivist Paradigm on the Development of Entrepreneurial Competencies in University Students ». Ghent, Ghent University, 245 p.

Jones, Caper. (662). 2008. *Applied Software Measurement: Global Analysis of Productivity and Quality*, Third. McGraw-Hill Osborne Media.

Kay, Robin. 2011. « Evaluating learning, design, and engagement in web-based learning tools (WBLTs): The WBLT Evaluation Scale ». *Computers in Human Behavior,* vol. 27, n° 5, p. 1849-1856.

Knox, Grahame 2009. « 40 Ice breakers for small groups ». p. 21. < http://insight.typepad.co.uk/40_icebreakers_for_small_groups.pdf >.

Larkin, Helen, and Ben Richardson. 2012. « Creating high challenge/high support academic environments through constructive alignment: student outcomes ». *Teaching in Higher Education,* vol. 18, n° 2, p. 192-204.

Leedy, Paul, and Jeanne Ellis Ormrod. 2010. *Practical research: planning and design*, Ninth edition. Pearson.

Lindsey, Lee, and Nancy Berger. 2009. « Experiential approach to instruction ». In *Instructional-Design Theories and Models*, sous la dir. de Reigeluth, Charles, and Alison Carr-Chellman. Vol. III, p. 117-142. New York: Routledge, Taylor & Francis Group

Lupton, Mandy. 2012. « Reclaiming the art of teaching ». *Teaching in Higher Education,* vol. 18, n° 2, p. 156-166.

McAuliffe, Garrett, and Karen Eriksen. 2011. *Handbook of Counselor Preparation : Constructivist, Developmental, and Experiential Approaches*. California: SAGE publications, Inc.

Network, HandsOn. 2006. « Team Building Exercises for College Students ». p. 4. < http://www.nationalserviceresources.org/files/BP_TeambuildingExercises_2010_HON.pdf >. Consulté le 22 December 2013.

Nunnally, Jum, and Ira Bernstein. 1994. *Psychometric theory*, Third. Coll. « McGraw-Hill Series in Psychology ». New York: McGraw-Hill 793 p.

Okoli, Chitu, and Suzanne D. Pawlowski. 2004. « The Delphi method as a research tool: an example, design considerations and applications ». *Information & Management,* vol. 42, n° 1, p. 15-29.

Pelech, James, and Gail Pieper (215). 2010. *The Comprehensive Handbook of Constructivist Teaching*. Charlotte, NC: Information Age Publishing, INC.

Pritchard, Alan , and John Woollard (106). 2010. *Psychology for the Classroom: Constructivism and Social Learning*. London and New York: Routledge, Taylor & Francis group.

Rainer, Austen, and Tracy Hall. 2003. « A quantitative and qualitative analysis of factors affecting software processes ». *Journal of Systems and Software,* vol. 66, n° 1, p. 7-21.

Reigeluth, Charles;, and Alison; Carr-Chellman. 2009. « Situational Principles of Instruction ». In *Instructional-design theories and models*, sous la dir. de Reigeluth, Charles;, and Alison; Carr-Chellman. Vol. III, p. 57-71. New York: Routledge, Taylor & Francis Group

Romiszowski, Alexander. 2009. « Fostering skill development outcomes ». In *Instructional design theories and models*, sous la dir. de Reigeluth, Charles, and Alison Carr-Chellman. Vol. III, p. 199-224. New York: Routledge, Taylor & Francis Group.

SAGE Research Methods. 2013. « Encyclopedia of Survey Research Methods ». < http://srmo.sagepub.com/view/encyclopedia-of-survey-research-methods/SAGE.xml >.

Salazar, Danny, Mónica Villavicencio, Verónica Macías and Monique Snoeck. 2004. « Estudio estadístico exploratorio de las empresas desarrolladoras de software asentadas en Guayaquil, Quito y Cuenca ». In *Jornadas Argentinas de Informática e Investigación Operativa 2004* (Córdoba, Argentina, 20 al 24 de Septiembre de 2004), 33 Edition. < http://www.cs.famaf.unc.edu.ar/33JAIIO/ >.

Stamelos, I., I. Refanidis, P. Katsaros, A. Tsoukias, I. Vlahavas and A. Pombortsis. 2000. « An Adaptable Framework for Educational Software Evaluation ». *Decision Making: Recent Developments and Worldwide Applications*. Vol. 45, p. 347-360. < http://dx.doi.org/10.1007/978-1-4757-4919-9_23 >. Consulté le October 10, 2013.

Staples, Mark, and Mahmood Niazi. 2008. « Systematic review of organizational motivations for adopting CMM-based SPI ». *Information and Software Technology,* vol. 50, n° 7-8, p. 605-620.

Suskie, Linda A. 2009. *Assessing student learning : a common sense guide : JB-Anker Series*.

The Centre for Teaching Excellence. 2013. « Teaching problem-solving skills ». < https://uwaterloo.ca/centre-for-teaching-excellence/teaching-resources/teaching-tips/developing-assignments/cross-discipline-skills/teaching-problem-solving-skills >. Consulté le July 2013.

The Joint Task Force on Computing Curricula Association for Computing Machinery IEEE-Computer Society. 2013. « Computer Science Curricula 2013, Ironman Draft (Version 1.0) ». p. 376. < http://ai.stanford.edu/users/sahami/CS2013/ironman-draft/cs2013-ironman-v1.0.pdf >.

Trienekens, J. J. M., R. J. Kusters, M. J. I. M. van Genuchten and H. Aerts. 2007. « Targets, drivers and metrics in software process improvement: results of a survey in a multinational organization ». *Software Quality Journal,* vol. 15, n° 2, p. 135-153.

Trudel, Sylvie. 2012. *Exercice pratique de mesure*. 4 p.

Villavicencio, M., and A. Abran. 2011a. « Educational Issues in the Teaching of Software Measurement in Software Engineering Undergraduate Programs ». In *Joint*

Conference of the 21st Int'l Workshop on Software Measurement and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA). (Nara, Japan, 3-4 Nov. 2011), p. 239-244.

Villavicencio, Monica, and Alain Abran. 2011b. « Facts and Perceptions Regarding Software Measurement in Education and in Practice: Preliminary Results ». *Journal of Software Engineering and Applications* vol. 4, n° 4, p. 227-234.

Villavicencio, Mónica, and Alain Abran. 2010. « Software Measurement in Software Engineering Education: A Comparative Analysis ». In *International Conferences on Software Measurement IWSM/MetriKon/Mensura*. (Stuttgart, Germany, 10-12 November, 2010), p. 633-644. Shaker Verlag.

Weber, Charles, and Beth Layman. 2002. « Measurement Maturity and the CMM: How Measurement Practices Evolve as Processes Mature ». Vol. 4, n° 3, p. 15. < http://www.compaid.com/caiinternet/ezine/layman-CMM.pdf >.

Webster, Allen L. 2000. *Estadistica aplicada a los negocios y a la economia*, Tercera. Santa Fe de Bogota: Irwin McGraw-Hill, 640 p.

Yazbek, Hashem. 2010. « Metrics Support in Industrial CASE Tools ». *Software Measurement News: Journal of the Software Metrics Community* (Magdeburg). August 2010, p. 40.

Zuse, Horst. 1998. *A framework of software measurement*. Berlin: Walter de Gruyter & Co.