Software Quality Engineering – where to find it in Software Engineering Body of Knowledge (SWEBOK)

Witold Suryn[1], Anabel Stambollian[2], Jean-Charles Dormeux[3], Luc Bégnoche [4]

[1]Software and Information Technology Engineering Dept, École de technologie supérieure. 1100 Notre Dame St. West, Montréal, Québec, H3C 1H3 Canada. E-mail: witold.suryn@etsmtl.ca

[2]Master student. Software and Information Technology Engineering Dept, École de technologie supérieure. 1100 Notre Dame St. West, Montréal, Québec, H3C 1H3 Canada. E-mail: anabel.stambollian.1@ens.etsmtl.ca

[3]Master student. Software and Information Technology Engineering Dept, École de technologie supérieure. 1100 Notre Dame St. West, Montréal, Québec, H3C 1H3 Canada. E-mail: jcdormeux@hotmail.com

[4]Master student. Software and Information Technology Engineering Dept, École de technologie supérieure. 1100 Notre Dame St. West, Montréal, Québec, H3C 1H3 Canada. E-mail: luc.begnoche.1@ens.etsmtl.ca

Abstract

Software quality engineering makes its way to wider professional recognition mostly through the specialized research and related publications; however, due to the dispersed nature of this process none of these efforts can create the visibility of the subject comparable to this of the Software Engineering Body of Knowledge (SWEBOK). As software quality makes the intrinsic part of SWEBOK, software quality engineering should be represented in this document in at least similar manner, allowing for easy recognition and equally easy application. The objective of the research discussed in this article was to verify the presence of software quality engineering in SWEBOK, evaluate the maturity and completeness of its representation and propose eventual improvements or additions.

1. Introduction

As part of this presented research, the latest version of SWEBOK [1] had been analyzed from the perspective of the core processes constituting the practices of software quality engineering. The main goal of such an analysis was to evaluate each Knowledge Area (KA) constituting SWEBOK in order to verify the level of representation of the subject of software quality engineering in this most prominent document of software engineering domain. The core processes of software quality engineering were identified through the analysis of Software Life Cycle processes published in the ISO standard ISO/IEC 12207:1995 [2] supported by the results of the research on Software Engineering Principles and Software Quality Implementation Model.

2. Abbreviations, terms and definitions

Within the text of this article the following abbreviations and definitions are applied:

SWEBOK – Guide to Software Engineering Body of Knowledge,

SQIM – Software Quality Implementation Model,

SLC – Software Life Cycle,

KA – Knowledge Area (in SWEBOK),

Internal quality - The totality of attributes of a product that determine its ability to satisfy stated and implied needs when used under specified conditions [3]

External quality - The extent to which a product satisfies stated and implied needs when used under specified conditions [4]

Quality in Use - The extent to which a product can be used by specified users to meet their needs to achieve specified goals with effectiveness, task efficiency safety and satisfaction in a specified context of use [5]

Operational quality – the extent to which a massively deployed software product satisfies stated and implied needs in terms of reliability and serviceability. Measurement and evaluation of operational quality requires valid statistical data representations [6].

3. Research methodology

The dedicated analysis methodology has been developed in order to realize the research objective presented previously. To keep both the research and its results as generic as possible, the hypotheses, terms and definitions used in this research are referenced to published and recognized literature sources, mainly standards issued by Subcommittee 7 (SC7) – System and Software Engineering of the International Organization for Standards (ISO). As the basis for validating the basic processes of software quality engineering domain, two ISO standards were taken into consideration: ISO/FCD 15288 - Information Technology - Life Cycle Management - System Life Cycle Processes [7] and ISO/IEC 12207-1995 - Information Technology – Software Life Cycle Processes [2], with the latter finally chosen as more generic.

The methodology consists of four phases, presented graphically in Fig.1:

Phase 1: Analyze, validate and if necessary, add the basic processes of software quality engineering definitions to the Software Life Cycle (SLC) processes and activities identified in ISO/IEC 12207,

Phase 2: Analyze, validate, adjust (if necessary) and map the set of basic processes of software engineering definitions identified in ISO/IEC 12207 with the processes and activities described in respective Knowledge Areas (KA) of SWEBOK,

Phase 3: Apply and asses the results of the mapping between the processes of software quality engineering identified in phase 1 with the processes and activities described in KAs of SWEBOK (phase 2),

Phase 4: Conclude the analysis and propose modifications to SWEBOK, if required.

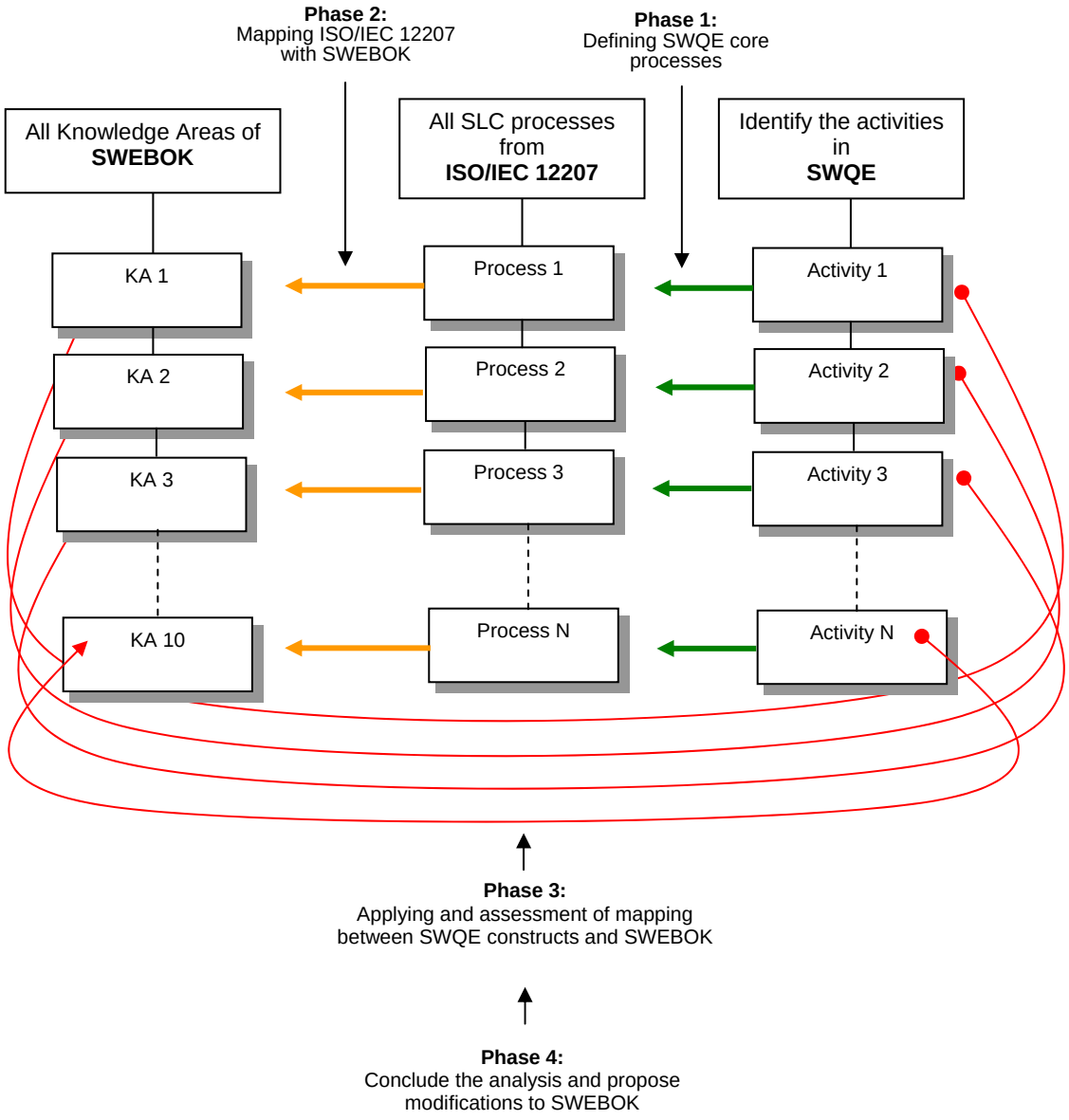

Fig. 1: The structure of the research methodology

4. Research execution and obtained results

The realization of the research and obtained results will be discussed following the order of the execution of phases of the methodology used for the project.

4.1. Phase 1: Analysis, validation and addition of the constructs of software quality engineering applying ISO/IEC 12207

The diagrams below illustrate the identification, definition and addition of processes of software quality engineering through the mapping to the Primary and Supporting Life Cycle Processes of ISO/IEC 12207.

The mapping approach applies the following two premises as starting points:

- the results of the research on Software Engineering Principles [8] published by Bourque et al. in 2002. The applied principle states (direct quote):

  Principle 7: Manage quality throughout the life cycle as formally as possible

- the definition of software quality engineering [6] published by Suryn in 2002 that states:

  The application of a continuous, systematic, disciplined, quantifiable approach to the development and maintenance of quality of software products and systems; that is, the application of quality engineering to software.

The direct implication of applying the above two premises allowed for building the corresponding software quality engineering-related life cycle model baptized in [6] Software Quality Implementation Model (SQIM) used as the reference model in mapping to ISO/IEC 12207 SLC processes (Fig.2 and 3).

Fig. 4 presents the proposed enhancements to the overall model of SLC processes from ISO/IEC 12207:1995

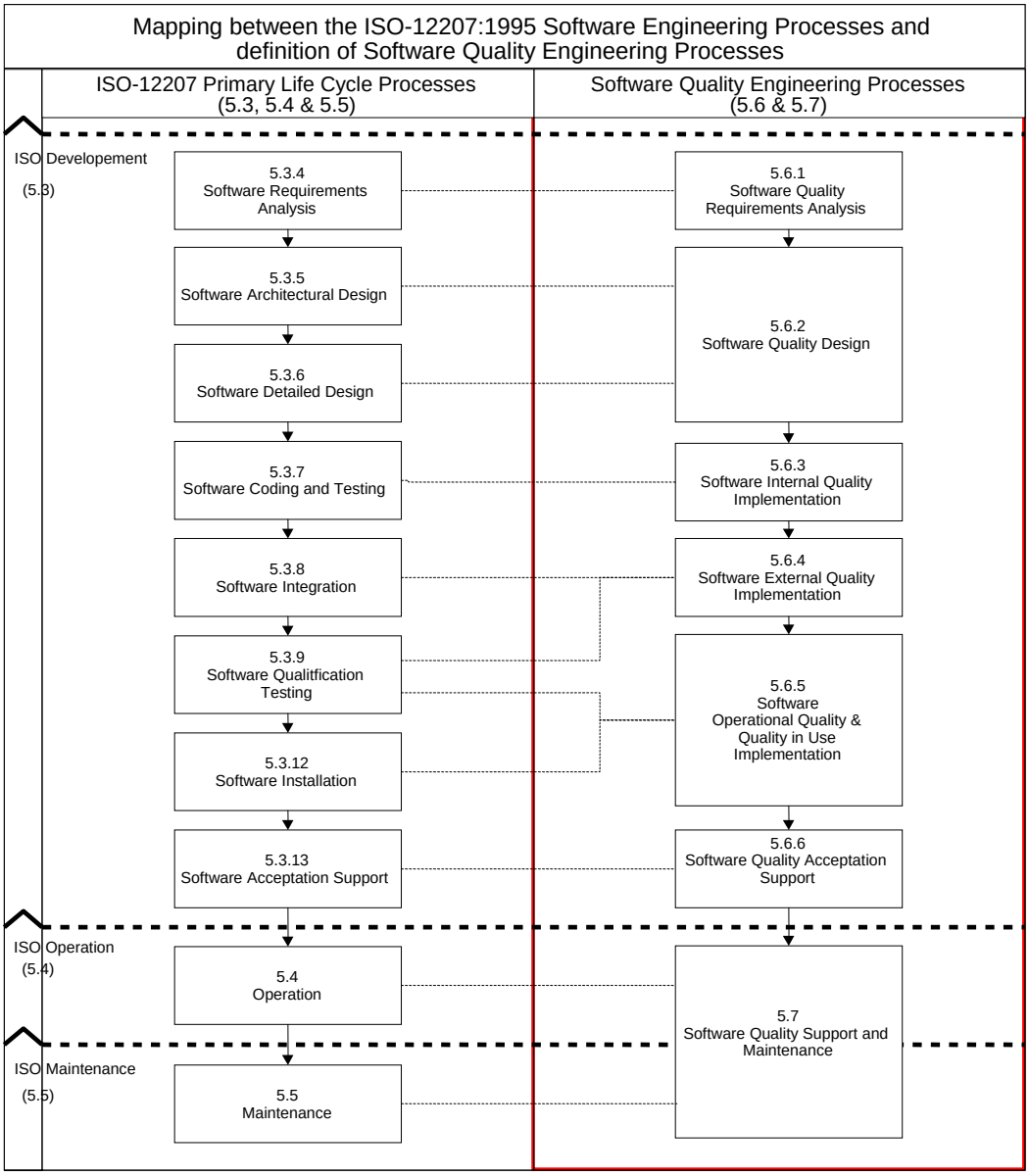

Fig. 2 : Mapping of Software Quality Engineering Constructs to the Primary Life Cycle Processes of ISO/IEC 12207

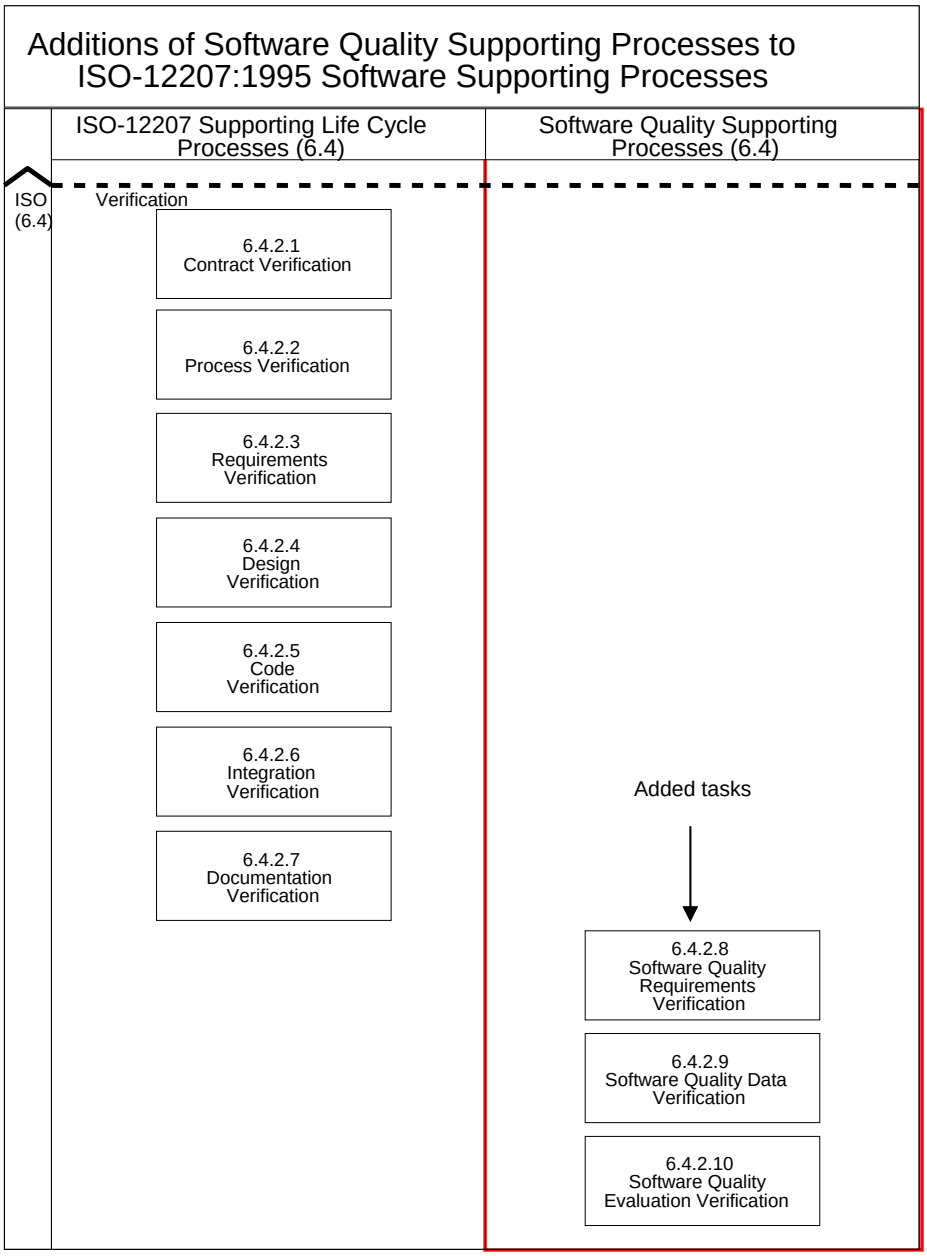

Fig. 3: Tasks added to ISO/IEC 12207 Verification Life Cycle Supporting Process

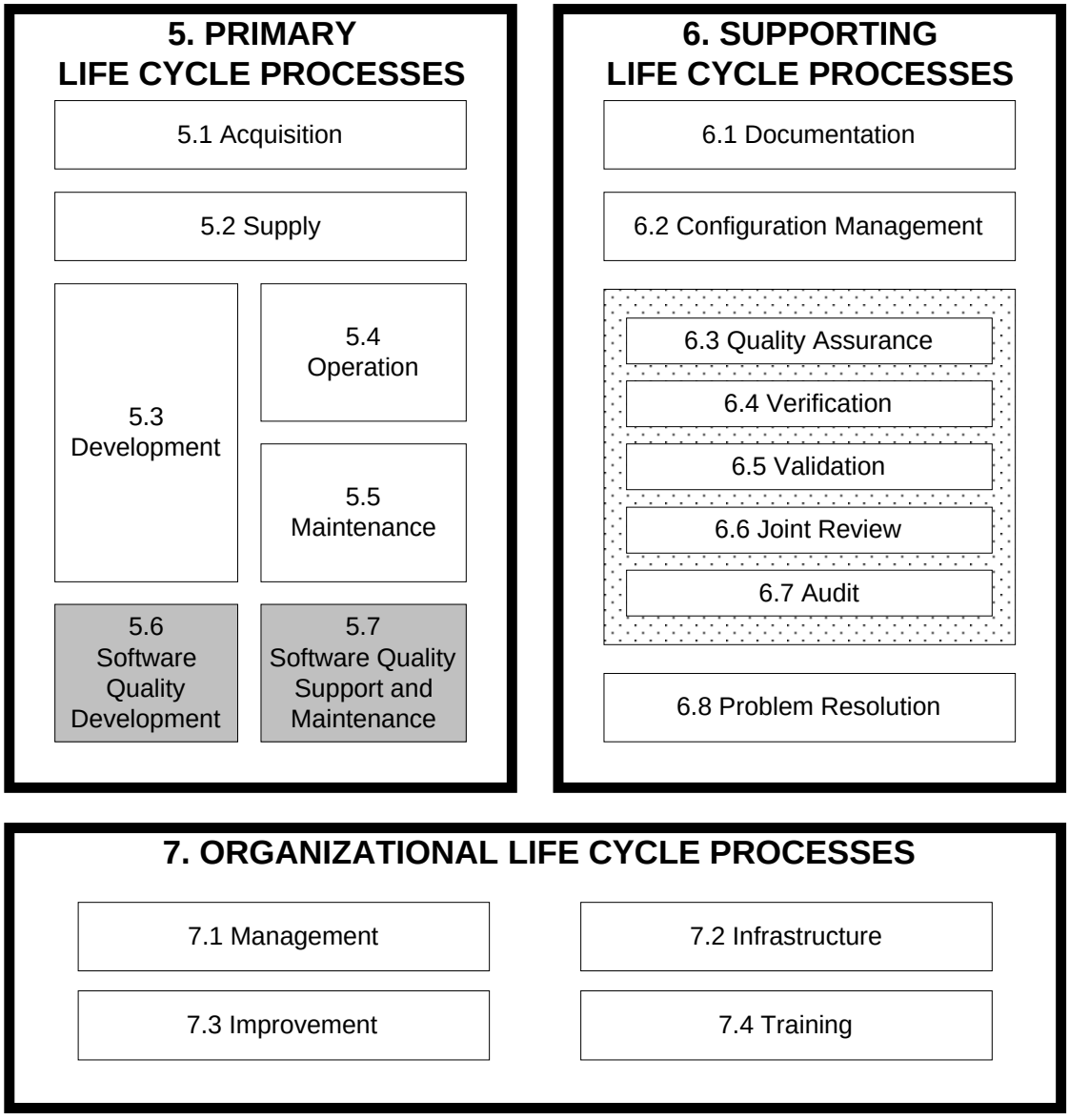

Fig. 4: Proposed additions to the overall ISO/IEC 12207 structure (5.6 & 5.7)

The following section presents the textual enhancements to ISO/IEC 12207 corresponding to the additions proposed in Fig.2, 3 and 4.

In this section, it has been decided to keep the original notation of ISO/IEC 12207 clauses in order to allow for easy identification of added parts against these existing within the original document. This notation form also helps in differentiating between new parts that are entirely added and those that were modified. The numbering used below corresponds to clauses of ISO/IEC 12207, not to numbering applied to clauses of this article.

5 Primary Life Cycle Processes

5.6 Software Quality Development Process

5.6.1 Software Quality Requirements Analysis

For each software item, this activity consists of the following tasks:

5.6.1.1 The quality engineer shall establish and document operational quality and quality in use requirements. These requirements shall be extracted from stakeholder needs and software requirements (functional and non-functional).

5.6.1.2 The quality engineer shall evaluate the operational quality and quality in use requirements considering the criteria listed below. The results of the evaluation shall be documented.

a) Traceability to stakeholder needs and software requirements;
b) External consistency with stakeholder needs and software requirements;
c) Internal consistency;
d) Testability and measurability;
e) Feasibility of building operational quality and quality in use in the software item.

5.6.1.3 The quality engineer shall establish and document the quality model for quality in use as described below.

a) Quality in use characteristics (and sub-characteristics if applicable);
b) Quality in use measurement primitives mapped to characteristics;
c) Derived measures and their measurement functions;
d) Base measures and their measurement methods or test scenarios;
e) Rating levels and target values for measurement primitives.

5.6.1.4 The quality engineer shall define and document tests that will be conducted to validate operational quality requirements.

5.6.1.5 The quality engineer shall develop an evaluation plan to control operational quality and quality in use requirements validation.

5.6.1.6 The quality engineer shall conduct joint review(s) in accordance with 6.6. Upon successful completion of the review(s), a baseline for the operational quality and quality in use requirements and evaluation plan shall be established.

5.6.1.7 If external and internal quality requirements are directly extractable from the stakeholder needs and the software requirements (functional and non-functional), the quality engineer shall refer to 5.6.2.

5.6.2 Software Quality Design

For each software item, this activity consists of the following tasks:

5.6.2.1 The quality engineer shall establish and document external and internal quality requirements. These requirements shall take into account quality in use requirements and stakeholder needs.

5.6.2.2 The quality engineer shall evaluate the external and internal quality requirements considering the criteria listed below. The results of the evaluation shall be documented.

a) Traceability to quality in use (stakeholder needs and software requirements if applicable);
b) External consistency with quality in use (stakeholder needs and software requirements if applicable);
c) Internal consistency;
d) Testability and measurability;
e) Feasibility of building external and internal quality in the software item.

5.6.2.3 The quality engineer shall establish and document the quality model for external and internal quality as described below.

a) External and internal quality characteristics;
b) External and internal quality sub-characteristics;
c) External and internal quality measurement primitives mapped to sub-characteristics;
d) Derived measures and their measurement functions;
e) Base measures and their measurement methods or tests;
f) Rating levels and target values for measurement primitives.

5.6.2.4 The quality engineer shall develop an evaluation plan to control external and internal quality requirements verification.

5.6.2.5 The quality engineer shall conduct joint review(s) in accordance with 6.6. Upon successful completion of the review(s), a baseline for the external and internal quality requirements and evaluation plan shall be established.

5.6.2.6 The quality engineer shall ensure that the software architectural design (according to 5.3.5) of the software item complies with external quality requirements. Particularly, the quality engineer shall assist the preliminary test requirements definition as described in 5.3.5.5 and shall participate to joint review(s) as described in 5.3.5.7.

5.6.2.7 The quality engineer shall ensure that the software detailed design (according to 5.3.6) of the software item complies with internal quality requirements. Particularly, the quality engineer shall assist the test requirements definition as described in 5.3.6.5 and shall participate to joint review(s) as described in 5.3.6.8.

5.6.3 Software Internal Quality Implementation

For each software item, this activity consists of the following tasks:

5.6.3.1 The quality engineer shall gather internal quality data by performing the tasks described below. The internal quality data shall be archived.

a) Gathering results from tests performed in 5.3.7.2;

b) Applying measurement methods and functions defined in the quality model of 5.6.2.3;

5.6.3.2 The quality engineer shall evaluate the internal quality data gathered considering the criteria listed below. The results of the evaluation shall be documented.

a) Traceability to internal quality requirements;

b) Internal consistency;

c) Repeatability, reproducibility, impartiality and objectivity.

5.6.3.3 The quality engineer shall establish a preliminary external and internal quality evaluation report based on the quality model defined in 5.6.2.3. This evaluation report is preliminary because it only contains interpretations of internal quality data.

5.6.3.4 The quality engineer shall establish improvement recommendations based on the preliminary evaluation report. The quality engineer shall ensure that these recommendations are implemented.

5.6.3.5 The quality engineer shall participate to joint review(s) of software code to ensure that it complies with internal quality requirements.

5.6.4 Software External Quality Implementation

For each software item, this activity consists of the following tasks:

5.6.4.1 The quality engineer shall gather external quality data by performing the tasks described below. The external quality data shall be archived.

a) Gathering results from tests performed in 5.3.8.2 and 5.3.9.1;

b) Applying measurement methods and functions defined in the quality model of 5.6.2.3;

5.6.4.2 The quality engineer shall evaluate the external quality data gathered considering the criteria listed below. The results of the evaluation shall be documented.

a) Traceability to external quality requirements;

b) External consistency with internal quality data;

c) Internal consistency;

d) Repeatability, reproducibility, impartiality and objectivity.

5.6.4.3 The quality engineer shall update the external and internal quality evaluation report defined in 5.6.3.3. This evaluation report is final because it contains interpretations of both external and internal quality.

5.6.4.4 The quality engineer shall conduct joint review(s) in accordance with 6.6. Upon successful completion of the review(s), a baseline for the external and internal quality evaluation report of the software item shall be established.

5.6.4.5 The quality engineer shall establish improvement recommendations based on the final evaluation report. The quality engineer shall ensure that these recommendations are implemented: If changes are recommended, the quality engineer shall re-enter the 5.6.3 and 5.6.4 phases, to ensure that these recommendations are implemented and to assist the software engineering team in their 5.3.7 and 5.3.8 phases

5.6.4.6 The quality engineer shall ensure that the integrated software (according to 5.3.8) complies with external quality requirements. Particularly, the quality engineer shall participate to joint review(s) as described in 5.3.8.6 and shall participate to audit(s) as described in 5.3.9.4.

5.6.5 Software Operational Quality & Quality in Use Implementation

For each software item, this activity consists of the following tasks:

5.6.5.1 The quality engineer shall gather quality in use data by performing the tasks described below. The quality in use data shall be archived.

a) Gathering results from tests performed in 5.3.9.1;

b) Applying measurement methods and functions defined in the quality model of 5.6.1.3;

5.6.5.2 The quality engineer shall evaluate the quality in use data gathered considering the criteria listed below. The results of the evaluation shall be documented.

a) Traceability to quality in use requirements;

b) External consistency with external quality data;

c) Internal consistency;

d) Repeatability, reproducibility, impartiality and objectivity.

5.6.5.3 The quality engineer shall establish the quality in use evaluation report based on the quality model defined in 5.6.1.3.

5.6.5.4 If the product environment permits it, the quality engineer shall ensure that the tests defined in 5.6.1.4 are conducted to evaluate operational quality. If not, the quality engineer shall ensure that the tests defined in 5.6.1.4 are conducted to evaluate operational quality in phase 5.6.6. The quality engineer shall establish the operational quality evaluation report based on the results of those tests.

5.6.5.5 The quality engineer shall conduct joint review(s) in accordance with 6.6. Upon successful completion of the review(s), a baseline for the quality in use evaluation report and operational quality evaluation report of the software item shall be established.

5.6.5.5 The quality engineer shall establish improvement recommendations based on both evaluation reports. The quality engineer shall ensure that these recommendations are implemented.

5.6.5.6 The quality engineer shall ensure that the software to qualify (according to 5.3.9) complies with operational quality (if the product environment permits it only, else it shall be ensured in phase 5.6.6) and quality in use requirements. Particularly, the quality engineer shall participate to audit(s) as described in 5.3.9.4.

5.6.5.7 The quality engineer shall ensure that the software installation, the software installation techniques and tool conform to the external quality, quality in use and operational quality (and internal quality if applicable) requirements.

5.6.6 Software Quality Acceptation Support

For each software item, this activity consists of the following tasks: conform

5.6.6.1 The quality engineer shall support the acquirer's acceptance review and testing of the software product considering external quality, quality in use and operational quality (and internal quality if applicable) . The results of the acceptance review and testing shall be documented.

5.7 Software Quality Support and Maintenance process

5.7.1 Operational testing

This activity consists of the following tasks:

5.7.1.1 The quality engineer shall assist in the development of the operational test plan as described in 5.4.1.1. to ensure that the operational quality requirement are respected.

5.7.2 Problem and modification analysis

This activity consists of the following tasks:

5.7.2.1 The quality engineer shall analyze the problem report or modification request for its impact on software quality requirements. If any impact is identified, the quality engineer shall assist the maintainer in his tasks described in 5.5.2.2 to 5.5.2.5.

5.7.3 Modification implementation

This activity consists of the following tasks:

5.7.3.1 The quality engineer shall enter the Software Quality Development Process (5.6) to assist the modification implementation described in 5.5.3.

5.7.4 Migration

This activity consists of the following tasks:

5.7.4.1 The quality engineer shall enter the Software Quality Acceptation Support Phase (5.6.6) to assist the migration described in 5.5.5.

6 Supporting Life Cycle Processes

6.4 Verification Process

6.4.2 Verification

6.4.2.8 **Software Quality Requirements Verification.** The software quality requirements shall be verified considering the criteria listed below:

a) The software quality requirements are consistent, feasible, testable, measurable and traceable to software requirements (or stakeholder needs if applicable);

b) The operational quality requirements are clearly distinguishable and separated from other quality requirements.

c) The quality in use requirements are clearly distinguishable and separated from other quality requirements.

d) The external and internal quality requirements are clearly distinguishable and separated from other quality requirements.

e) The quality models implements correctly characteristics, sub-characteristics (if applicable), measurement primitives, derived measures (if applicable), base measures, measurement functions (if applicable) rating levels and target values.

6.4.2.9 **Software Quality Data Verification.** The software quality data shall be verified considering the criteria listed below:

a) The software quality data is consistent, repeatable, reproducible, impartial, objective and traceable to software quality requirements;

b) The operational quality data is clearly distinguishable and separated from other quality data.

c) The quality in use data is clearly distinguishable and separated from other quality data.

d) The external quality data is clearly distinguishable and separated from other quality data.

e) The internal quality data is clearly distinguishable and separated from other quality data.

f) The software quality data contains test results and measured values (if applicable).

g) The software quality data gathering tasks have been performed in accordance with an evaluation plan.

6.4.2.10 **Software Quality Evaluation Verification.** The software quality evaluation shall be verified considering the criteria listed below:

a) The software quality evaluation are consistent with the rating levels, target values and measured values (or test results if applicable);

b) The quality improvement recommendations are traceable to the corresponding software quality evaluations.

4.2. Phases 2 and 3: Mapping of ISO/IEC 12207 SLC and software quality engineering processes to SWEBOK

The results of phases 2 and 3 of the research are presented in an aggregated form, where the recommended additions to ISO/IEC 12207 addressing the software quality engineering have already been inserted into the main document. In Table 1, the numbers in cells indicate the mapping of ISO/IEC 12207 to SWEBOK KAs with these new, related to software quality engineering requiring full or partial addition in ***Bold Italic***. The "X" in the cell indicates that the process identified in ISO/IEC 12207 is fully represented in SWEBOK.

SWEBOK / ISO/IEC 12207	Software Requirements	Software Design	Software Construction	Software testing	Software Maintenance	Software Configuration Management	Software Engineering Management	Software Engineering Process	Software Engineering Tools and Methods	Software Quality
5.1 Acquisition process							X			
5.2 Supply process							X			
5.3 Development process	5.3.4	5.3.5, 5.3.6	5.3.7, 5.3.8	5.3.7, 5.3.8, 5.3.9						
5.4 Operation process										
5.5 Maintenance process					X					
5.6 Software Quality Development Process	***5.6.1, 5.6.2***	***5.6.2***	***5.6.3, 5.6.4***	***5.6.3, 5.6.4, 5.6.5, 5.6.6***						***X***
5.7 Software Quality Support and Maintenance process					***X***					***X***
Documentation process	X	X	X	X	X	X	X	X	X	X
Configuration management process						X				
Quality assurance process										X
Verification process	6.4.2.3, ***6.4.2.8***	6.4.2.4	6.4.2.5, 6.4.2.6, ***6.4.2.9, 6.4.2.10***	6.4.2.2, ***6.4.2.9, 6.4.2.10***						***X***
Validation Process										X
Joint review process										X
Audit process										X
Problem resolution process					X					
Management process							X			
Infrastructure process									X	
Improvement process								X		
Training process	X	X	X	X	X	X	X	X	X	X

Tab. 1: Mapping of enhanced set of ISO/IEC 12207 SLC Processes to SWEBOK KAs

4.3. Phase 4: Propositions of modifications to SWEBOK

Similarly to clause 4.1, the following section presents the enhancements to SWEBOK in textual form, corresponding to the additions proposed in Table 1.

The text in ***Bold Italic*** indicates the domains or sub-domains that are suggested to be added in order to fill up the identified omissions. The domains that are not discussed in the following text are considered as well addressing the subject of software quality engineering. In case of omissions identified on the level of principal taxonomy, the textual additions are enriched by graphical representation, with proposed enhancements, also in ***Bold Italic***.

Chap 2 : Software Requirements	
1. Software Requirements Fundamentals	Noted deficiencies
1.3 Functional and Nonfunctional Requirements	- The title should include Quality requirement like this: "Functional, Nonfunctional & Quality Requirements" - Shall differentiate quality requirements from non-functional requirements. - Shall define quality requirements categories like this: ▪ operational quality requirements, ▪ quality in use requirements, ▪ external quality requirements, ▪ internal quality requirements.
2. Requirements Process	Noted deficiencies
2.2 Process Actors	- May add software quality engineers in the typical examples of software stakeholders. - Shall add that the software quality engineer's job is to negotiate trade-offs concerning software quality requirements.
2.4 Process Quality and Improvement	- Shall clarify what this topic covers: process quality or software quality. The reader may misinterpret quality as software quality requirements.
3. Requirements Elicitation	Noted deficiencies
3.1 Requirements Sources	- Shall include the role of the software quality engineer in all software requirements collection (in 3.1 and 3.2).
3.2 Elicitation Techniques	

4. Requirements Analysis	Noted deficiencies	
4.1 Requirements Classification	- Shall include software quality requirements subtypes in classification.	- Shall add this concept in all sub sections: that software quality requirements may be derived or influenced by stakeholder needs, system requirements and software requirements.
4.2 Conceptual Modeling	- Shall add software quality models in the types of models.	
4.3 Architectural Design and requirements	- Shall add that software quality engineers must ensure that the architectural design complies with the software quality requirements.	
4.4 Quality Modeling	- ***Shall elaborate on how the quality models are refined and serve as tools to analyze quality requirements.***	
5. Requirements Specification	Noted deficiencies	
5.1 The system definition document	- Shall add the notion of quality in use.	
5.3 Software requirements specification	- All of the quality requirements should be take into account.	
6. Requirements validation	Noted deficiencies	
6.4 Acceptance tests	- Shall add that the software quality engineer must develop an evaluation plan to control operational quality and quality in use requirements validation.	

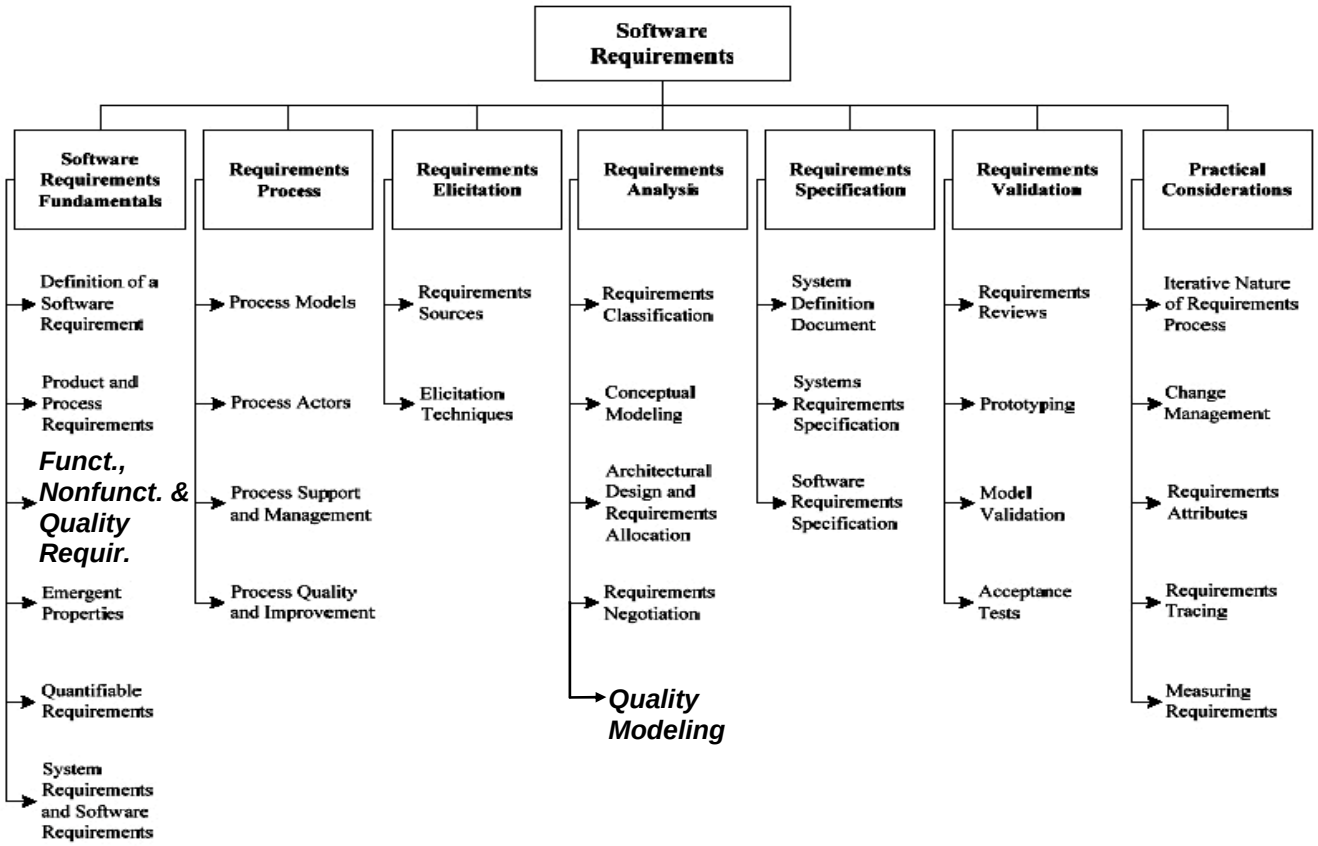

Fig. 5: Breakdown of topics for Software Requirements KA

Chap 3 : Software Design	
4. Software Quality Design Analysis and Evaluation	Noted deficiencies
4.1 Quality attributes	- "Various attributes are generally considered important for obtaining a software design of good quality…": The quality attributes should be presented as a contribution to the software quality and not only as the software design quality.
4.2 Quality analysis and evaluation techniques	- Shall specify that tracing to quality requirements is also essential.

Chap 4 : Software Construction		
1. Software Construction Fundamentals	Noted deficiencies	
1.2 Anticipating change	/	
1.3 Constructing for verification	- Shall include the notion of traceability to quality requirements. It will support the quality requirement verification.	
2 Managing Construction	Noted deficiencies	
2.1 Construction models	/	- Shall include the setting of quality objectives for each deliverable included in all of the managing construction activities.
2.2 Construction planning	/	
2.3 Construction measurement	- Shall include a reference to quality measurements specific to artifacts produced during the software construction.	
3. Practical Considerations	Noted deficiencies	
3.6 Construction quality	- Shall include that internal quality plays an important role in software development and the construction of quality: "Construction quality activities are differentiated from other quality activities by their focus. Construction quality activities focus on code and on artifacts that are closely related to code: small-scale designs-as opposed to other artifacts that are less directly connected to the code, such as requirements, high level designs, and plans."	

Chap 5 : Software Testing	
3. Test Techniques	Noted deficiencies
3.1 Based on tester's intuition and experience	- Shall include the quality requirements testing in all sub-clauses
3.2 Specification based	
3.3 Code based	
3.4 Fault based	
3.5 Usage based	
3.6 Based on nature of application	
3.7 Selecting and combining techniques	

Chap 6 : Maintenance
- The definition of that knowledge area is consistent and general enough so that no noted deficiencies need to be presented

Chap 7 : Software Configuration Management
- The definition of that knowledge area is consistent and general enough so that no noted deficiencies need to be presented

Chap 8 : Software Engineering Management	
2 Software Project Planning	Noted deficiencies
2.6 Quality Management	- Must explain that the software quality engineer is responsible for establishing and documenting preliminary software quality models. These preliminary models should contain only top-level quality characteristics from a management and user-oriented perspective. Moreover, these models should contain some direct metrics (see IEEE 1061-1998) and threshold that will be used to validate these top-level quality characteristics. - It must be clear that during planning the software quality engineer should not be concerned by technical and design-oriented perspectives and consequently, should not have to establish complete software quality models prior to planning.
6. Software Engineering Measurement	Noted deficiencies
6.1 Establish and sustain measurement commitment	- Could include the software quality engineering activities as a good example of a measurement process; should explain that the whole topic is applicable to software quality engineering but is not restricted to.
6.2 Plan the measurement process	
6.3 Perform the measurement process	
6.4 Evaluate measurement	

Chap 9 : Software Engineering Process	
4 Process and Product Measurement	Noted deficiencies
4.1 Software Products Measurement : 4.1.3 Quality Measurement	- Because this section is generic and because it refers to more detailed sections, nothing has to be modified.

Chap 10 : Software Engineering Tools & Methods	
1 Software Engineering Tools	Noted deficiencies
1.9 Software Quality Tools	- A new category of tools could be created: Software Quality Assessment Tools. Such a category would overlap with software requirements tools (1.1), test evaluation tools (1.4), performance analysis tools (1.4), and measurement tools (1.7). However, it seems that such a category will emerge sooner or later according to discussions about GDQA (Graphical Dynamic Quality Assessment) and IGQ (Integrated Graphical Assessment of Quality)[1].

Chap 11 : Software Quality	
2 Software Quality Management Processes	Noted deficiencies
2.4 Software Quality Measurement	- This subtopic is originally under Practical Considerations. - This subtopic should be moved under Software Quality Management Processes for clarity .The reason is that it is directly derived from topic 6 (Software Engineering Measurement) described in Software Engineering Management KA. It should also refer directly to this last topic.

[1] Buglione, L.; Kececi, N.; Abran, A., AN INTEGRATED GRAPHICAL ASSESSMENT FOR MANAGING SOFTWARE PRODUCT QUALITY, in 12 International Conference on Software Quality, Ottawa, Ontario, 2002, pp. 14.

3 Software Quality Engineering	Noted deficiencies	
3.1 Quality Definition and Design	- ***This new topic should cover how the quality models must be used and refined to prepare quality evaluation. It should cover the tasks 5.6.1, 5.6.2 and 6.4.2.8 in the proposed ISO12207.*** - ***It should refer to KAs Software Requirements and Software Design. It should also refer to Models and Quality Characteristics from this KA.***	- ***This new topic is intended to cover the software quality engineer role and activities. We propose only 2 subtopics because almost everything as be described in other KAs. This is intended to be a résumé that will clearly distinguish Software Quality Engineering from the overall Software Engineering.***
3.2 Quality Construction and Evaluation	- ***This new topic should cover how software construction is an iterative process that involves quality construction and quality evaluation. It should cover the tasks 5.6.3, 5.6.4, 5.6.5, 5.6.6, 6.4.2.9 and 6.4.2.10 in the proposed ISO12207.*** - ***It should refer to KAs Software Construction and Software Testing. It should also refer to Verification & Validation and Software Quality Measurement from this KA.***	

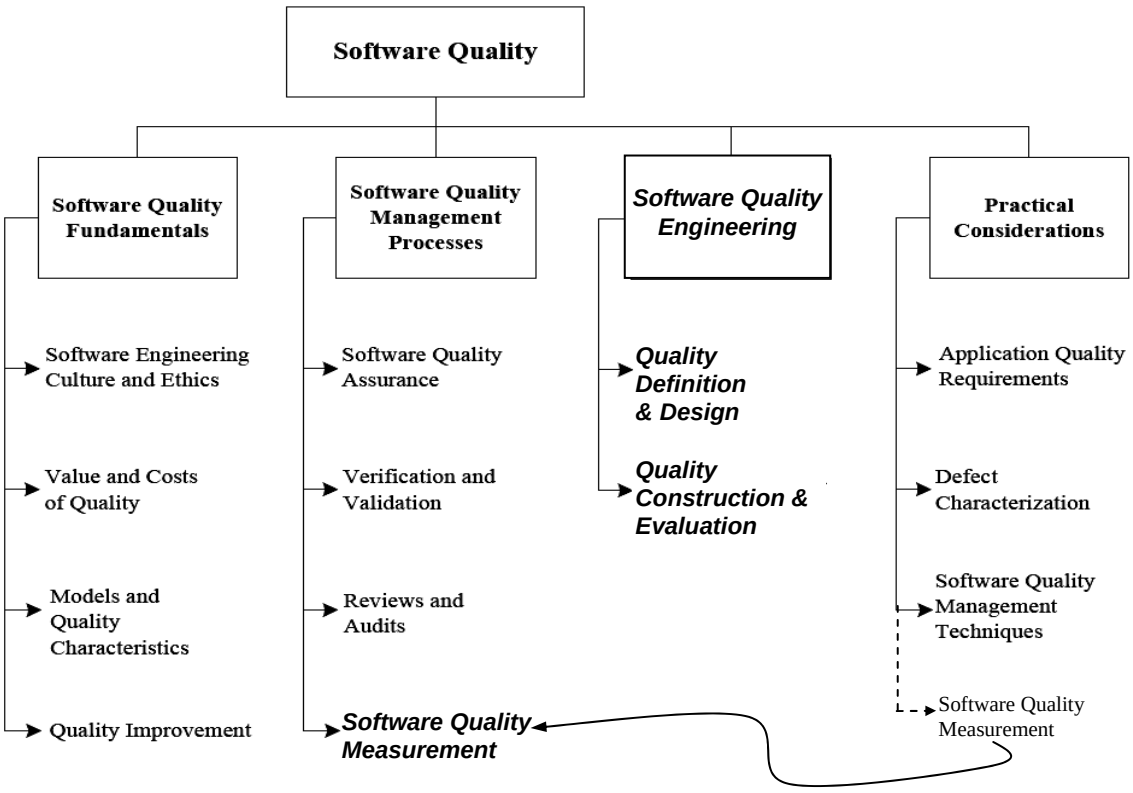

Fig. 6: Proposed Breakdown of topics for the Software Quality KA

Chap 12 : Related Disciplines of Software Engineering
- The definition of that knowledge area is consistent and general enough so that no noted deficiencies need to be presented

5. Conclusion and Future Work

The research program presented in this article had as the principal objective to verify the level of representation of software quality engineering practices and processes in the most prominent document of the software engineering domain: Guide to Software Engineering Body of Knowledge (SWEBOK). According to the results,, several missing elements were identified, and consequently, modifications and additions were proposed. The continuation of this research is foreseen in the form of direct co-operation with SWEBOK editorial committee, both on IEEE-CS and ISO levels. This unique opportunity could have become possible due to the recent decision of re-opening the SWEBOK project with the project management center located again at École de technologie supérieure (ETS) in Montreal, Canada.

References

1 *Guide to the Software Engineering Body of Knowledge*, Version 2004, SWEBOK®. A project of the IEEE Computer Society Professional Practices Committee, http://www.swebok.org

2 ISO/IEC 12207-1995, *Information Technology – Software Life Cycle Processes*

3 ISO/IEC 9126, *Software engineering –Product quality – Part 3: Internal metrics*

4 ISO/IEC 9126, *Software engineering –Product quality – Part 2: Externa metrics*

5 ISO/IEC 9126, *Software engineering –Product quality – Part 4: Quality in use metrics*

6 Suryn, W., Notes de cours de « *SYS867-Ingénierie de la qualité* », Session AUT-2005, École de Technologie Supérieure

7 ISO/FCD 15288 - *Information Technology - Life Cycle Management -System Life Cycle Processes*

8 Bourque P., Dupuis R., Abran A., Moore J.W., Tripp L., Wolff S., *Fundamental Principles of Software Engineering – A Journey*. Journal of Systems and Software 2002

9 Stambollian, A., Dormeux, J-C., Begnoche, L. *Analyse de SWEBOK du point de vue de l'ingénierie de la qualité du logiciel, V1.2*, Paper written in the context of a master degree course at the École de Technologie Supérieure, given by Witold Suryn, 2005