An ISO/IEC standards-based quality requirement definition approach: Applicative analysis of three quality requirements definition methods

Rachida Djouab, Witold Suryn
École de technologie supérieure, Montréal, Québec, Canada

Abstract

It is known in the industry that software quality requirements engineering is still an immature discipline since its absence results in dissatisfied users and costly applications. The identification and specification of software quality requirements from system and user requirements is becoming a prominent task in software engineering. The lack of these requirements or their inappropriate identification may compromise business processes and may impact negatively the results of any development project. The presented paper discusses three quality engineering approaches which address quality requirements. The main objective of this research study is to define a methodology for building ISO/IEC standards-based quality approach for quality requirements identification.

Introduction

Software quality engineering requires new practices and methods. In particular, quality requirements identification methods are the focus of this paper. In the last years, some methodologies have been proposed for dealing with the quality requirements [1, 2, 3, 8, 9, 10, 11, 12, 13, and 14]. In all of them, the key point is the difficulty to identify quality requirements and represent them in models or processes. Quality requirements are often difficult to check and usually associated to non-functional requirements. This is partly due to the lack of a consensus of structured descriptions related to software quality requirements. This paper points out the relevant methods & techniques addressing the quality requirements. The obtained research study will serve as basis for implementing the software product quality requirements identification process using ISO/IEC 9126 quality standard as framework.

The paper is structured as follows: in section 2, each method will be described according to the following points (table 1): a) description of the method, b) activities and process of this method, c) requirements addressed by this method, d) quality model used by this method and tools/techniques supporting this method. After that, analysis and discussion of these methods will be done by establishing their advantages and drawbacks (table 3). In particular, the difficulty of identifying quality requirements from user and system requirements is highlighted. Section 3 describes the improved methods with additional activities for identifying quality requirements, new techniques for extracting these quality requirements from user and system requirements and templates for describing these quality requirements. Based on these improvements, a summary of issues on the new quality approach design and the future research are presented in section 4. The choice of methods has been based on the diversity of analyzed methods in order to capture as large as possible approach to quality requirements identification and definition. The obtained research study will serve to propose the new quality requirement engineering approach.

Method M1 "Requirement model for quality attributes"

Description of the method: This method defines a process to identify and specify quality attributes that crosscut requirements and to integrate them into the functional requirements at early stage of the software development process:

1. It proposes a template to specify quality attributes at the requirement stage (table 2);
2. Extends "Use Cases" and sequence diagrams [1, 17 and 18] to specify the integration of those attributes with functional requirements.

Activities of this method: The model of process is compatible with UML formalism [1 and 15] and is composed of three important activities (figure 1):

1. Identification of system requirements and selection of quality attributes relevant to the stakeholder's and application domain from those requirements;
2. Specification of requirements:
   a. Specify functional requirements by using "Use Case" based approach [17];
   b. Describe quality attributes by using templates and specify quality attributes crosscutting functional requirements.
3. Integration of crosscutting quality attributes with functional requirements.

Table 1 – Quality requirements methods

Method	Description	Quality model Used	Activities	Requirements	Tools/techniques
M1 : Requirement model for quality attributes	Integration of quality attributes into the functional requirements	Template to specify the quality attributes at the requirement stage (see example)	Identify use cases & quality attributes Specify functional requirements &quality attributes Identify crosscutting quality attributes Integrate crosscutting quality attributes with functional requirements	Functional and quality of the software.	UML models: use cases, sequence & class diagrams Special templates

1

M2 : SHEL oriented requirement engineering	Integration of functional, cognitive & quality requirements	SHEL model [1]	Analysis the work system of the specific environment Identify user needs & critical issues Define requirements & architecture of the SHEL system Define the functional, cognitive & quality requirements Design prototypes & Mock-ups	Functional, cognitive & quality of the system	Observations Interviews Heuristic analysis Video recording Task allocation Design models Design patterns Simulation
M3 : Prometheus	Goal oriented method that integrates quantitative & qualitative approaches to quality control of SPL (software product lines)	Combination of several models: fixed & customized models	Define quality goals Specify quality characteristic s (model content) Specify relationships Review the model Operatio-nalize the model	Quality: quality goals & characteristic s of the software product.	Measurement Goal Template (MGT) GQM [7], Interviews, Questionnaire s BBN (Bayesian Belief Nets), Monte Carlo, Weighted average, Analytica, Hugin, Netica, MSBNx

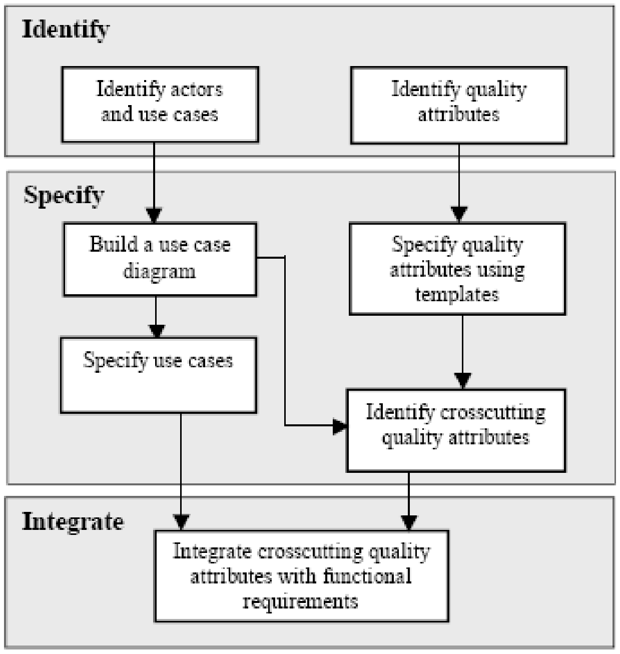

Identify
Identify actors and use cases
Identify quality attributes
Specify
Build a use case diagram
Specify quality attributes using templates
Specify use cases
Identify crosscutting quality attributes
Integrate
Integrate crosscutting quality attributes with functional requirements

Figure 1: A requirements model for quality attributes [1]

Requirements addressed by this method: The requirements addressed by these method are of two types: functional and quality. The quality requirements are specified as "quality attributes" and are defined as "global properties of a system, assumptions, constraints or goals of stakeholders".

Quality model used by this method: the quality model used by this method is a template for describing quality attributes (table 2)

Tools/techniques supporting this method: The techniques used in this method are: "Use Case" approach (UML model, sequence & class diagrams) for specifying functional requirements and templates for describing quality attributes.

Table 2 – Template for describing quality attributes [1]

Name	The name of the quality attribute.
Description	Brief description.
Focus	A quality attribute can affect the system (i.e. the end product) or the development process.
Source	Source of information (e.g. stakeholders, documents).
Decomposition	Quality attributes can be decomposed into simpler ones. When all (sub) quality attributes are needed to achieve the quality attribute, we have an AND relationship. If not all the sub quality attributes are necessary to achieve the quality attribute, we have an OR relationship.
Priority	Expresses the importance of the quality attribute for the stakeholders. A priority can be MAX, HIGH, LOW and MIN.
Obligation	Can be optional or mandatory.
Influence	Activities of the software process affected by the quality attribute.
Where	List of models (e.g. sequence diagrams) requiring the quality attribute.
Requirements	Requirements describing the quality attribute.
Contribution	Represents how a quality attribute can be affect by the others quality attributes. This contribution can be positive (+) or negative (-).

Method M2 "SHEL oriented requirement engineering"

Description of the method: This method applies an approach which addresses integration issues of different types of requirements from the hardware, software and

[1] The SHEL model provides an integrated view by considering any productive process or activity performed by a combination of Hardware, Software and Liveware resources within a specific

liveware resources within a specific environment. Acronym SHEL means: S: « Software »; H: « Hardware »; E: « Environment » and L: « Liveware ». The SHEL model supports a systemic view which supports definition of different types of requirements related to system (software and hardware) and human aspects (human roles, interaction, and help in breakdown-situations).

Activities of this method: The definition process of these requirements is composed of 6 phases (Figure 2):

1. **Work analysis:** is a profound analysis of the work system of the specific environment.
   a. It studies the way in which the productive process is performed taking into account all the resources that contribute and interact in the process execution.
   b. The knowledge required to perform a productive process is captured by using techniques as: observations, interviews, video recording, checklist, etc.
   c. It produces models and processes as tools supporting process performance, objects in the work process, interactions and social & work practices in order to describe the existing work system with its critical issues and weaknesses.
2. **Identification of user needs and critical issues:** The focus changes to the weakness of the actual system by eliciting critical issues dues to the knowledge distribution among the SHEL resources and their interaction.
   a. It provides a basis for alternative design considerations represented by various prototypes and design models early in the design process.
   b. It figures out suitable knowledge distributions for an effective use of the resources.
3. **Definition of SHEL requirements and Design of the SHEL system:** All the collected information contributes to defining the requirements and architecture of the system according to the SHEL model.
4. **Definition of functional, cognitive and quality requirements**. Quality requirements are defined according to approach UCD ("User Centered Design") [2 and 4] which is based on the user viewpoint to define quality needs for the system. A quality model is defined with quality characteristics by using task analysis technique [2 and 4]. The tool (SQUID) [2, 4 and 20] for data acquisition is used to control and evaluate software quality.
5. **Design of prototypes and Mock-ups:** The requirements are mapped into "design patterns" represented in prototypes, mock-ups, design models and scenarios.
6. **Validation by experts of the work environment:** The last phase of one iteration cycle in SHEL oriented requirements engineering approach is the validation by the domain experts:
   a. System compliance with requirements is evaluated. Evaluation takes into account "measurable criteria" such as performance and criteria such as usability, cognitive workload and level of cognitive support;
   b. Requirements and projected system are validated in the real system environment;
   c. The prototypes are used with user involvement to validate possible modifications;
   d. The "mock-ups" and prototypes are used with a set of predefined scenarios to ease the communication and feedback with users and to anticipate bad design solutions.

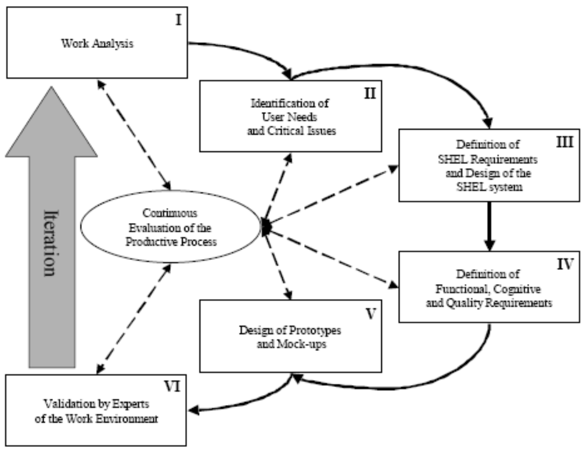

I
Work Analysis
II
Identification of User Needs and Critical Issues
III
Definition of SHEL Requirements and Design of the SHEL system
Iteration
Continuous Evaluation of the Productive Process
IV
Definition of Functional, Cognitive and Quality Requirements
V
Design of Prototypes and Mock-ups
VI
Validation by Experts of the Work Environment

Figure 2: Oriented SHEL process for defining requirements [2]

Requirements addressed by this method: The requirements addressed by this method are of 3 types: functional, cognitive and quality. The quality requirements describe the user needs representing the design and end user views. The "work analysis" phase is useful to identify quality characteristics and quality requirements. User viewpoint is used to refine characteristics of the quality model.

Quality model used by this method: ISO/IEC 9126 standard [5] is used as quality model. It consists of important quality characteristics for the final product. Quality sub characteristics and attributes refine the quality model. They can be internal or external quality attributes.

Tools/techniques supporting this method: The techniques used in this method are:

1. Observations, interviews, heuristic analysis, video recording , checklist for capturing knowledge of the productive process in the "work analysis " phase;
2. Prototypes and design models in the "Identification of user needs and critical issues" phase;
3. Design patterns, prototypes, mock-ups, scenarios in the "Design of prototypes and Mock-ups" phase;

4. Prototypes, mock-ups, scenarios in the “Validation by experts of the work environment” phase.

Method M3 “Prometheus to model quality in SPL (Software Product Lines)”

Description of the method: This method is a requirements quality modeling approach combining characteristics of various quality models to meet requirements of flexibility, reuse and transparency.

Activities of this method: The definition process of these quality requirements is composed of three phases:

1. **Requirements specification phase:** during this phase a quality model is developed. It is composed of the following activities (figure 3):
   a. **Define quality goals:** quality goals are defined by the system users and other stakeholders by applying the measurement goal template (table 3). The goal formulation is conducted iteratively and serves as baseline for the evaluation stage. The template also describes other quality modeling like object (artifact) and stakeholder (viewpoint).
   b. **Specify quality characteristics (content of model)** describes the refinement of quality goals into quality characteristics and sub characteristics. Refining quality characteristics is done by organizing interview sessions with domain experts using as support questionnaires, case studies and quality models.
   c. **Specify relationships (structure of model).** Here, two types of relationships are defined: **decomposition,** which specifies decomposition of high quality characteristics into detailed sub characteristics, and **influence**, that defines which sub characteristic influences the value of other characteristics.
   d. **Review the model:** the model is reviewed according to the implementation feasibility;
   e. **Operationalize the model:** the model is quantified (characteristics and relationships) and qualified by applying the BBN technique (Bayesian Belief Network) [3 and 21].
2. **Application phase:** During this phase, the model is used to evaluate the requirements;
3. **Packaging phase:** Information on acquired experience during application of the model is collected in order to improve it and to reuse it in other projects.

Table 3 – Measurement Goal Template [3]

Dimension	Definition	Example
Object	Which artefact is analysed?	*Analyze* the software system
Purpose	Why is the object analyzed?	*for the purpose of* evaluation
Quality Focus	Which characteristics of the object are analyzed?	*with respect to* maintainability
Viewpoint	Who will use the data collected?	*from the viewpoint of* the system user
Context	In which environment does the analysis take place?	*in the context of* company X and development project Y

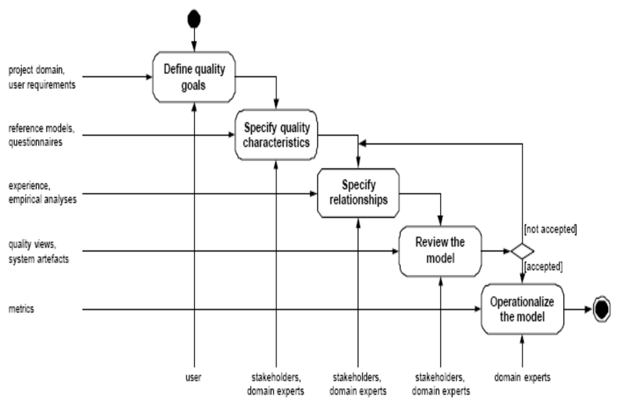

project domain, user requirements
reference models, questionnaires
experience, empirical analyses
quality views, system artefacts
metrics
Define quality goals
Specify quality characteristics
Specify relationships
Review the model
Operationalize the model
[not accepted]
[accepted]
user
stakeholders, domain experts
stakeholders, domain experts
stakeholders, domain experts
domain experts

Figure 3: Activities during the specification phase of the Prometheus approach [3].

Requirements addressed by this method: the requirements addressed by this method are quality goals and characteristics of the software product.

Quality model used by this method: various quality models are combined and used by this method.

Tools/techniques supporting this method: Techniques used in this method are: GQM (Goal Question Metric) [7] to define quality goals, interviews and questionnaires with domain experts to refine quality goals into quality characteristics and sub characteristics, SQUID tool [3 and 20] for modeling and evaluating software quality and BBN technique [21] for quantification of relationships as well as for integration of quantitative and qualitative data within the quality model. BBN technique also provides efficient mechanism for combining measures into quality estimates. Tools support can be: Analytica, Hugin, MSBNx and Netica.

Analysis and discussion

According to Table 1, method M1 "Requirement model for quality attributes" [1] defines a process to identify and specify quality attributes that crosscut requirements including their integration with the functional requirements [1]. This method uses templates to specify quality attributes [1] and is investigating other approaches such as ATAM (Architecture Tradeoff Analysis Method), composition patterns & goal oriented requirements engineering related to quality attributes and crosscutting concerns [1]. It proposes a **new way**: "aspect-oriented paradigm" to integrate quality requirements (non functional requirements) with the functional requirements [22].

Method M2 "SHEL oriented requirements engineering" [2] deals with integration of different types of requirements defined for software, hardware and liveware resources. It is supported by the SHEL model [2] that provides an integrated view by defining a combination of hardware, software and liveware resources within a specific environment. The method uses an integrated requirements engineering approach to define quality requirements [2]. This method defines requirements by a systemic requirements engineering process.
Method M3 « Prometheus approach » [3] is a goal-oriented method that integrates quantitative and qualitative approaches to software product lines (SPL) quality modeling. Prometheus combines several models (fixed quality models and customized quality models) to define quality requirements [3].

Prometheus enables to [3]:

- Start quality evaluation early in the development process;
- Learn effectively across several product variances/releases;
- Refine the quality model through subsequent projects
- Integrate quantitative (measured based) and qualitative approaches;
- Combine different contexts of software quality individual views (as developers, users) and evaluation objects as (processes, products, resources).

The following table 4 summarizes advantages/drawbacks of these quality approaches.

Table 3– Advantages/drawbacks of quality approaches

Quality requirements methods	**Advantages**	**Drawbacks**
M1: Requirement model for quality attributes	It proposes a **new way**: "aspect-oriented paradigm" to integrate quality requirements (non functional requirements) with the functional requirements[22]	There is a lack of model or process to identify quality requirements from system/user requirements; The template for quality attributes did not mention anywhere how to trace quality attributes to quality requirements; There is no mention how to specify quality requirements; It is not indicated how ISO 9126 is used to specify quality characteristics and sub-characteristics.
M2: SHEL method	It uses **an integrated** requirements engineering approach to define quality requirements	It is not mentioned anywhere in this method how to define "quality in use"; The lack of techniques to extract quality requirements from SHEL requirements and design of the SHEL system; It is not focused on the mapping activities of quality engineering instruments with the product definition phase at early requirements stage.
M3: Prometheus approach	It gave a **detailed description** of quality requirements identification activities; It integrates quantitative (measured based) and qualitative approaches.	There is no mention how to identify quality requirements from quality goals (defined by GQM method [7]); It did not use ISO/IEC9126 as quality standard; There is no mention how to specify quality requirements.

This section describes the extent to which these methods address quality requirements.

The method M1 using template for describing quality attributes is very interesting in the sense that knowledge about these attributes is collected (source, focus, decomposition, influence, requirements describing them, and their contribution to other attributes). In fact the template gives information about description of the quality attribute itself, its information source, its decomposition into sub attributes, its importance for the stakeholders, its influence on activities of the development process, the list of models requiring this quality attribute, requirements describing this quality attribute and the list of quality attributes that being able to affect it. However, it is not specified anywhere how to identify these quality attributes from system and user requirements in models or processes. Furthermore, the template gives information about the decomposition of quality attribute into sub attributes and the requirements describing it. But, it is not mentioned how these requirements are decomposed into quality attributes which is important for the identification step of quality attributes and for requirements traceability.

Method M2 describes quality requirements as user needs. These needs have been identified in the first "work analysis" phase of the SHEL oriented process. The third phase "Definition of SHEL requirements and Design of the SHEL system" defines the requirements and architecture of the system according to the SHEL model which represent the starting point for defining quality requirements and other requirements cited before (cognitive and functional).

This method defines all requirements (cognitive, functional and quality) from work analysis and synthesis of user needs but it seems to be costly and do not support all features required for an integrated method. It also does not indicate how to identify quality requirements from SHEL system and design architecture. In addition, It is not mentioned anywhere in this method how to define "quality in use" that represents quality viewpoint of the user, which is also important in the quality requirements identification step.

As mentioned before, method M3 "Prometheus" enables to start quality evaluation early in the development process. It uses GQM method to define quality goals. The goal formulation is conducted iteratively and serves as baseline for the evaluation step. They are defined by system users and other stakeholders related to the project are involved in acceptation of the evaluation. This approach defines its own quality model by organizing interview sessions with domain experts who contribute in definition of quality goals and quality characteristics. Sessions are supported by techniques as questionnaires, case studies and existing quality models.

This approach has the following advantages:

1. Starting quality evaluation early in the development process;
2. Learning effectively across several product variances/releases;
3. Integrating quantitative (measured based) and qualitative approaches;
4. Combining different contexts of software quality individual views (as developers, users) and evaluation objects as (processes, products, resources).
5. Applicable across different companies, to any project and incorporate views of all relevant project stakeholders;
6. Reuse of quality experience packaged in existing quality models across other projects. It also supports the reuse of measurement data as well as quality characteristics and their relationships;

However, this method does not use the ISO/IEC 9126 as quality model. In addition, it is not mentioned how the quality requirements are extracted from quality goals and how they are specified.

In summary, these three methods present these common limitations as:

1. Difficulty to identify quality requirements and represent them in models and processes;
2. Absence of clear guidelines about the way to provide a consensus view on quality characteristics & their relationships;
3. Inability to reuse quality experience from other projects and companies (except Prometheus)

Improved methods

This section presents improvements of the three previous methods based on ISO/IEC standards 9126. The main improvements are represented by activities, templates and techniques for extracting and describing quality requirements.

For ISO/IEC standards-based method M1 (see figure 4) [1]:

1. Added activities to the quality attributes definition process
    a. Identify quality requirements;
    b. Specify quality requirements by using templates (see table 3);
    c. Specify quality characteristics sub characteristics;
2. Template for quality requirements specification (see table 5).

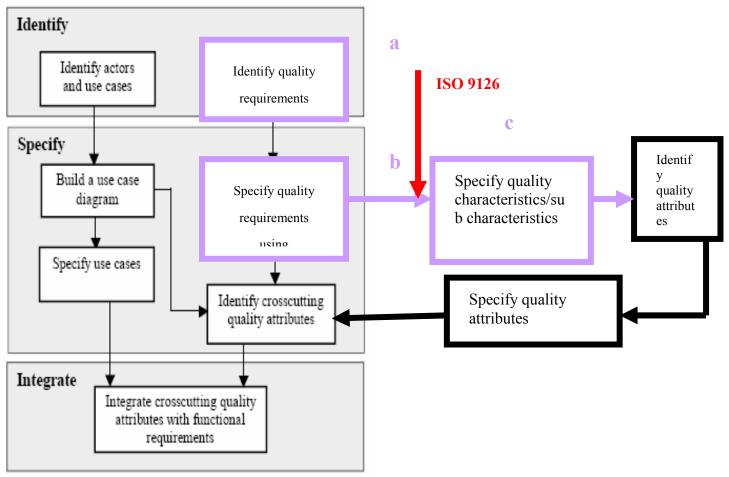

Identify
Identify actors and use cases
Identify quality requirements
a
ISO 9126
c
b
Specify
Build a use case diagram
Specify quality requirements using
Specify quality characteristics/sub characteristics
Identify quality attributes
Specify use cases
Identify crosscutting quality attributes
Specify quality attributes
Integrate
Integrate crosscutting quality attributes with functional requirements

Figure 4: ISO/IEC Requirement model for quality requirements

Table 5 – Template for quality requirements

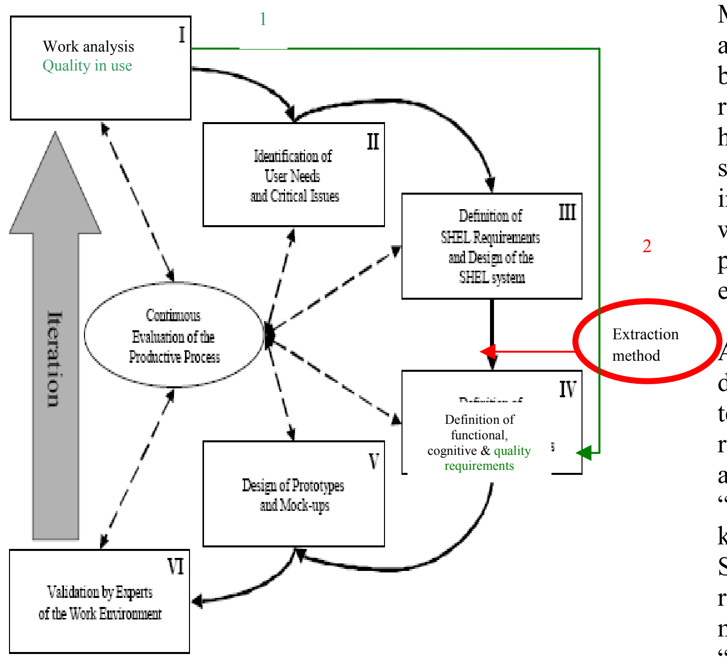
Requirements	**Definition**	**Example**
Name	The name of the quality requirement	Security
Description	Brief description	The system must: Protect the vehicle' owner registration data Guarantee integrity in the data transmitted to the bank Guarantee integrity on data changed/queried by the operator
Focus	The requirement can affect the end product or the development process.	System
Source	Source of information (stakeholders, documents)	Stakeholders
Decomposition	Quality requirements can be decomposed in characteristics & sub characteristics according to ISO 9126	Integrity & Confidentiality
Categorisation	Quality requirements can be categorised in quality in use, operational quality, external quality & internal quality	External Quality & Internal Quality
Priority	Express the importance of the quality requirement for the stakeholders, the priority can be MAX, LOW, & MIN	MAX
Obligation	Can be optional or mandatory	Mandatory
Influence	Activities of the software process & other categories of requirements (functional, non functional, cognitive) affected by the quality requirement	Design, system architecture & implementation Functional requirements (register Vehicle (integrity), Pay Bill (security))
Contribution	Represents how a quality requirement can be affected buy others quality requirements. This contribution can be (+) or negative (-).	(+) to functionality & portability (-) to usability & efficiency

For ISO/IEC standards-based method M2 (see figure 5) [2]:

1. Quality in use identification and definition task ;
2. Extraction method for quality requirements from SHEL requirements and design of SHEL system.

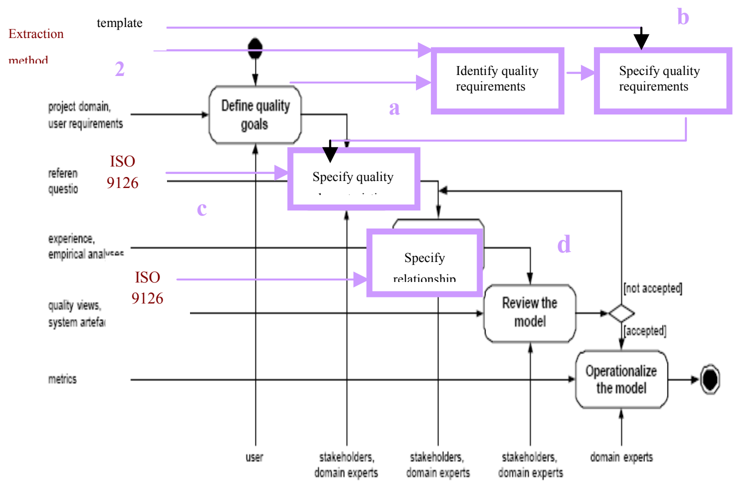
For ISO/IEC standards-based method M3 (see figure 6) [3]:

1. Added activities to the specification phase of Prometheus approach
   a. Identify quality requirements from quality goals;
   b. Specify quality requirements by using templates (see table 5);
   c. Specify quality characteristics sub characteristics;
   d. Specify decomposition and influence relationships among characteristics;
2. Extraction method of quality requirements from quality goals.



Work analysis
Quality in use
I
1
Identification of User Needs and Critical Issues
II
Definition of SHEL Requirements and Design of the SHEL system
III
2
Extraction method
Continuous Evaluation of the Productive Process
Iteration
Definition of functional, cognitive & quality requirements
IV
Design of Prototypes and Mock-ups
V
Validation by Experts of the Work Environment
VI

Figure 5 – ISO/IEC SHEL oriented quality requirements engineering approach



Extraction method
template
2
b
project domain, user requirements
Define quality goals
a
Identify quality requirements
Specify quality requirements
referen questio
ISO 9126
Specify quality
c
experience, empirical analyses
Specify relationship
d
ISO 9126
quality views, system artefa
Review the model
[not accepted]
[accepted]
metrics
Operationalize the model
user
stakeholders, domain experts
stakeholders, domain experts
stakeholders, domain experts
domain experts

Figure 6: Activities for quality requirements specification phase

Conclusion

This paper presents and discusses three methods addressing quality requirements. Method M1 describes a model for quality attributes specification and focuses on quality attributes that crosscut functional requirements. However it does not indicate how to select quality attributes relevant to the application domain and stakeholders and how to identify these quality attributes from system requirements.
Method M2 defines all requirements (cognitive, functional and quality) from work analysis and synthesis of user needs but it seems to be costly and does not support all features required for an integrated method. It also does not indicate how to identify the quality requirements from the SHEL system and Design architecture. Method M3 proposes an interesting approach to identify quality characteristics without applying ISO/9126 as quality standard. It is particularly dedicated for projects performing early evaluation in the SPL (software product lines) context.

Analysis and discussion of these methods led us to propose different improvements (techniques, activities and templates) contributing to identification of quality requirements and to definition of the proposed ISO/IEC approach. Based on these improvements, it can be said that "ISO/IEC standards-oriented SHEL" method requires more knowledge about the definition of quality in use in the SHEL process. The "ISO/IEC standards-oriented requirement model for quality attributes" method requires more activities for quality requirements identification. The "ISO/IEC standards-based Prometheus" is the most adapted and suitable for quality requirements definition since it gives important quality requirements identification activities in the speciation phase. In addition, its flexibility allows it to integrate ISO/IEC 9126 standard in its existing quality models. However, it requires some refinements as for example extraction and definition of quality requirements from quality goals.

The planned continuation of this research will focus on:

1. Developing quality requirements identification activities (based on Prometheus approach);
2. Implementing extraction techniques of quality requirements;
3. Developing decomposition techniques of quality requirements with respect to traceability mechanism;
4. Certifying quality requirements identification process.

References

1. I. Brito, A. Moreira, and J. Araujo, "A Requirements Model for Quality Attributes". Workshop on "Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design", 1st International Conference on Aspect-Oriented Software Development, University of Twente, Enschede, Holland, 2002.
2. M. Felici, M.A. Sujan, and M. Wimmer, "Integration of Functional Cognitive and Quality Requirements: A Railways Case Study," ESCOM - SCOPE 2000, Rome, Italy, pp. 395-403, 2000.

3. A. Trendowicz and T. Punter, "Quality Modeling for Software Product Lines", 4th International Conference on Product Focused Software Process Improvement PROFES 2002, Rovaniemi, Finland, 2002
4. M. Felici, A. Pasquini, and S. De Panfilis (1998) "Software Quality in User-Centred Design," ESCOM-ENCRESS, Rome, Italy, pp. 239-247.
5. ISO/IEC 9126. Software Engineering - Product Quality – Part 1: Quality Model, ISO 2001
6. ISO/IEC 9126. Software Engineering - Product Quality – Part 2: External Metrics, ISO 2003
7. ISO/IEC 9126. Software Engineering - Product Quality – Part 3: Internal Metrics, ISO 2003
8. GRAY A. & MACDONELL S.G., GQM++ A Full Life Cycle Framework for the Development and Implementation of Software Metrics Programs, University of Otago, Dunedin, New Zealand, Technical Report, 1997
9. W. Suryn, A. Abran, "ISO/IEC SQuaRE. The 2nd generation of standard for quality of software product". Proceedings of 7th IASTED International Conference on Software Engineering and Applications, SEA 2003, November 3-5, 2003, Marina del Rey, CA, USA
10. Suryn W., Abran A., Laporte C., "An integrated life cycle quality model for general public market software products". Proceedings of 12th International SQM & INSPIRE Conference 2004, Canterbury, Kent, UK 5-7 April 2004.
11. Suryn W., Gil B., "ISO/IEC9126–3 internal quality measures: are they still useful?" Proceedings of HCI International 2005, 22-27 July 2005, Las Vegas, Nevada USA.
12. Suryn W., Girard D., Suryn-Abran Consolidated Quality Lifecycle (CQL) Model - the Applicative Evolution. BIS 2005, April 2005, Poznań, Poland
13. Suryn W., Kahlaoui A., Georgiadou E., Quality engineering process for the Program Design Phase of a generic software life cycle. 13th International SQM & INSPIRE Conference 2005, Gloucestershire, Cheltenham, UK
14. Suryn W., Hailey V. A., Coster A., "Huge potential user base for ISO/IEC 90003 – the state of the art for improving quality in software engineering". ISO Management System International No.4, July-August 2004
15. Suryn W., "Thoughts on Teaching Software Quality Engineering". Proceedings of 8th Annual INSPIRE Conference, April 23 – 25, 2003. Glasgow, Scotland, UK
16. Jacobson I., Booch G., and Rumbaugh J., (1998) "*Unified Modeling Language 1.3",* White paper, Rational Software Corp., 1998.
17. Jacobson I., Booch G., and Rumbaugh J., (1999) "*Unified Software Development Process",* Addison-Wesley.
18. [Jacobson92] Jacobson, I., et al. Object-oriented Software Engineering: A Use Case Driven Approach. Addison Wesley. Reading, MA.
19. Constantine L. (1997) "The case for Essential Use Cases" Object Magazine. SIGS Publications. NY, NY
20. B. Kitchenham, S. Linkman, A. Paquini, V. Nanni (1997) "The SQUID approach to defining a quality model", Software Quality Journal, Vol.6, pp.211-233.
21. N. Fenton, P. Krause and M. Neil (2001) "A probabilistic model for software defect prediction », accepted for publication IEEE Transactions Software Eng.
22. J. Araújo, P. Coutinho (2003) "Identifying Aspectual Use Cases Using a Viewpoint-Oriented Requirements Method", Early Aspects 2003: Aspect-Oriented Requirements Engineering and Architecture Design, Workshop of the 2nd International Conference on Aspect-Oriented Software Development, Boston, USA, 17 March 2003.