Applying ISO/IEC 9126-1 Quality Model to Quality Requirements Engineering on Critical Software

Motoei AZUMA
Department of Industrial and Management Systems Engineering
School of Science and Engineering
Waseda University
azuma@azuma.mgmt.waseda.ac.jp

ABSTRACT
In order to develop a software product for a critical system, specifying quality requirements is vitally important. Quality requirements should be defined based on various stakeholders’ needs. Software quality impacts the information system’s behavior, and the behavior impacts the behavior of the External-System that contains the information system. Safety is an issue of the External-System. A software product alone is harmless, because it can do nothing without computer hardware. However, any software quality characteristic, such as security and reliability, impacts the External-System’s safety. In this paper, conceptual models for quality requirements are presented. Then needs, requirements, and quality requirements are defined. Requirements for a Quality Requirements Engineering method are also stated. Then a method for Quality Requirements Engineering and associated specification is provided with a simple example.

Keywords
Quality requirements, ISO/IEC 9126, quality model, critical software

1. INTRODUCTION
In general, product quality has significant influence on developers, dealers and users of the product. For example, Y Food Company that caused large-scale food poisoning case was obliged to close its door. Software quality cannot be an exception. Especially, the quality of software products for mission critical systems has large impacts on its stakeholders, such as users and developers as well as the public.

In order to implement critical software quality, it is necessary that stakeholders’ needs are precisely analyzed and reflected in the requirements. One of the most important issues of requirements engineering is that the quality model for a target system and the quality model for the associated software cannot be the same. For example, in order to improve the safety of the target system, security, reliability, and usability of the software should be improved.

Though most of popular software requirements technologies support only functional requirements, it is extremely important that software quality requirements be clearly defined, especially for critical systems’ software [9, 10]. Quality requirements should be exhaustive for all quality characteristics within a quality model, because every quality characteristic of software product may have influence on system safety. Quality requirements should be also objective, accurate, and quantitative, and should also provide evaluation criteria [7].

A quality model is a very useful tool for quality requirement engineering as well as quality evaluation. ISO/IEC 9126-1 provides a software product quality model. It is intended to be used as a general purpose default standard quality model [3].

ISO/IEC JTC1/SC7/WG6 is developing ISO/IEC 25000 SQuaRE (Software Quality Requirements and Evaluation) series of international standards (IS), including new IS on Quality Requirements (25030), which is now in the 2nd CD stage [2]. WG6 decided to start to develop a new Quality Model (25010) as a revision of ISO/IEC 9126-1 Quality Model. WG6 is also planning to revise 9126-2, -3, and -4, External, Internal and Quality-In-Use Metrics respectively, as a part of the SQuaRE series of International Standards [4, 5, 6].

The purpose of this paper is to suggest an idea of how to use associated quality metrics for safety critical systems requirements engineering, to list issues, and to invite contributions of experts in order to make the SQuaRE series more useful. Contributions on metrics on safety and security are also expected.

The SQuaRE series themselves are general purpose. However they must be consistent with other communities’ standards, such as safety, security, reliability, dependability, and usability.

The concept and definitions of needs, requirements and quality requirements are stated in this paper. Then a method for defining quality requirements is proposed. The method is based on experiences but is not yet well validated.

2. CONCEPTUAL MODEL FOR QR
Figure 1 shows the relationships between system and software. An information system consists of computer and communication hardware devices and software products. An External-System consists of information systems, people, machines, buildings and other artifacts. Examples of External-Systems include business systems,
A problem is defined as two states, i.e. the current states and the goal states of an External-System, its components, human factors, as well as its environment. Current states are effects of the current system’s behavior, and goal states are expected effects of the proposed system that is defined by requirements.

However, as the desirable states are not always obvious and may be different depending on each stakeholder’s position, the “problem” is redefined using goal states. When the desirable states are precisely defined as goal states and a system that is supposed to achieve the goal states are defined as requirements specifications, it is the proposed solution. Goal states should represent stakeholders’ needs.

Figure 2 shows the relationships between the current system, proposed system and realized system. Horizontal lines represent the transition of systems and software, and vertical lines represent cause and effect relationships. For example, current software is a cause of the current information system’s behavior.

3. CONCEPTS AND DEFINITIONS

Needs

Needs for a product are expectations of stakeholders for the effects of the product when it is actually operated, which means such action to the software product as development, distribution, release, installation, use and maintenance. In this context, stakeholders include developers, salesmen, system managers, users, end users, and maintainers. Stakeholders have their own needs for a product depending on their own positions.
“Needs” are categorized into stated needs and implied needs. Some needs are implied either because a stakeholder thinks it is too obvious to actually state the needs, or because no one is aware of the needs.

Some needs are contradictory. For example, a novice user wants a software product that is easy to learn and use; on the other hand, experienced engineers may want a product that is fast to use with many functions. Therefore, needs should be identified and selected for each Context-Of-Use. These needs should then be transformed into requirements. In other words, requirements should be derived from the stakeholders’ needs. Figure 3 shows the relationships between needs, requirements and design.

In this figure, an arrow means a relationship of “transform to” and “derived from”. For example, “External-System Requirements” are derived from “Selected & Specified Stakeholders’ Needs”, and “External-System Requirements” are transformed into “External-System Design”.

**Requirements**

In order to clarify the relationships between needs and requirements, “requirements” is defined in this paper as follows.

Requirements: Requirements are the external specification of specific needs that a product is expected to satisfy.

SWEBOK describes software requirements as follows [11].

“At its most basic, a software requirement is a property which must be exhibited in order to solve some problem in the real world. Hence, a software requirement is a property which must be exhibited by software developed or adapted to solve a particular problem.”

Functional requirements describe the functions that the software is to execute; for example, formatting some text or modulating a signal.

Non-functional requirements are the ones that act to constrain the solution. Non-functional requirements are sometimes known as constraints or quality requirements. They can be further classified according to whether they are performance requirements, maintainability requirements, safety requirements, reliability requirements, or one of many other types of software requirements.

System requirements are the requirements for the system as a whole. In a system containing software components, software requirements are derived from system requirements.

Specified requirements do not always satisfy selected and specified needs. Therefore it is necessary that the specified requirements be validated so that the proposed system will satisfy the needs at the earliest possible stage of software development lifecycle using, for example, prototyping.
Software Quality Requirements

Information System’s requirements and design should be transformed into software quality requirements, i.e. Functional Requirements and Quality Requirements (Figure 3). In order to clarify the concept of quality requirements, the following definitions are applied.

**Quality**: the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs [3]

**Quality model**: the set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality.

When software quality is measured and evaluated by attributes of the software product itself, the quality is named as internal quality.

ISO/IEC 9126-1 defines a quality model that should be used as the default quality model. The model defines three types of software quality: Quality-In-Use, External Quality, and Internal Quality. They are defined as follows.

**Quality-In-Use**: the user’s view of the quality of the software product when it is used in a specific environment and a specific Context-Of-Use.

Quality-In-Use measures the extent to which users can achieve their goals in a particular environment, rather than measuring the properties of the software itself. The concept of Quality-In-Use by this definition is close to the concept of users’ needs.

**External Quality**: the totality of characteristics of the software product from an external view. It is the quality when the software is executed, which is typically measured and evaluated while testing in a simulated environment with simulated data using external metrics.

**Internal quality**: the totality of characteristics of the software product from an internal view. Internal quality is measured and evaluated against the internal quality requirements.

The ISO model consists of six internal and external quality characteristics. It also defines four Quality-In-Use Characteristics, i.e. Effectiveness, Productivity, Safety, and Customer Satisfaction. As Customer Satisfaction usually reflects all quality properties, the author modified the model a little. The modified quality model is shown in Figure 4. Three Quality-In-Use Characteristics written in italic characters are new.

Safety, which is Quality-In-Use Characteristic, is defined in ISO/IEC 9126-1 as;

**The capability of the software product to achieve acceptable levels of risk of harm to people, business, software, property or the environment in a specified Context-Of-Use.**

A software product itself is completely safe, because it can do nothing. It does something only when it is executed as a part of an information system. An information system
itself outputs only information, either correct or erroneous. Erroneous output information may affect the safety of the External-System. Every characteristic of external and internal software quality has some possibility of causing safety problem on the External-System. Therefore, internal quality does not include the safety characteristic in this quality model.

Based on these definitions, software quality requirements can be categorized into External Quality Requirements, Internal Quality Requirements, and Quality-In-Use Requirements. Each category of software quality requirements is defined as follows.

**External Quality Requirements** specify the required level of quality from the external view. They include requirements derived from user quality needs, including Quality-In-Use requirements.

**Internal Quality Requirements** specify the required level of quality from the internal view of the product. Internal quality requirements are used to specify properties of interim products, including static and dynamic models, other documents and source code.

**Functional Requirements** are requirements for algorithms that transform input to output. The same input may cause different system behavior based on the state of the system.

Functional requirements and functionality requirements are considered to be different. Functionality is one of six Quality Characteristics that ISO/IEC 9126-1 defines. It is defined as;

**Functionality**: The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.

Therefore, while **Functional Requirements** define all functions that are necessary to satisfy selected and specified needs, **Functionality Requirements** provide decision criteria that contribute to deciding the priority of each function when the software product is used under specific condition, in other word, Context-Of-Use.

4. **REQUIREMENTS FOR REQUIREMENTS ENGINEERING METHODOLOGIES**

**Requirements for a needs analysis method**

Based on the concepts and definitions stated above, requirements for a needs analysis method are as follows.

1. A method should be applicable to every type of stakeholder and should support solicitation not only of stated needs but also implied needs.

![Figure 4: Modified ISO/IEC 9126-1 Quality Model](image-url)
A method should support selecting a need from alternatives or harmonizing contradictory needs.

Selected needs should be specified as formally as possible, by using form sheets, screen form, or language such as XML.

Requirements for a Requirements Engineering Method
Based on the concepts and definitions stated above, requirements for a Requirements Engineering method are as follows.

(4) Requirements should be specified objectively and should provide criteria for validation.

(5) Requirements specifications should be easily understandable not only by requirements analysts but also other stakeholders, including software designers, users, and end users.

(6) Requirements specification should include not only functional requirements but also quality requirements, as well as constraints. Other information that influence the requirements, such as users' profile and the environments of the software, should also be stated.

Requirements for Quality Requirements Engineering Method
Based on the concepts, definitions, and requirements stated above, requirements for a Quality Requirements Engineering and Specification Method are as follows.

(7) Quality Requirements should be specified for all Quality Characteristics based on their criticality [8].

(8) Quality Requirements should be specified as formally and comprehensively as possible.

In this respect, use of a well designed form is helpful.

(9) Quality Requirements should be objective, and should provide measures and evaluation criteria.

(10) Quality Requirements should be reflected in the Functional requirements.

5. PROPOSED QR ENGINEERING METHOD
Outline of the Method
Quality requirements should be specified based on specified needs and functional requirements. The author proposed Quality-In-Use concept and related methodology at the 2nd International Conference on Software Quality [1]. The following is a Quality Requirements Engineering Method based on the same concept but improved it a little.

Recently, more attention has been focused on object oriented approaches than structured approaches. However, the structured approach has strength from the human cognitive capability aspect. Therefore, the proposed method combines the merits of both approaches. It consists of two activities, i.e. General Requirements Engineering and Detailed Requirements Engineering. It means that stakeholders can understand their own needs if they have an overview of the system.

General Requirements Engineering
The following is outline of the General Requirements Engineering method.

(1) State an outline of the target system in order to show it to the stakeholders and explain the purpose and outline of the target system for the purpose of soliciting their needs.

(2) Draw a first cut of the system overview diagram based on the outline statement. Use Case Diagrams and IDEF0 are useful for the purpose. However, as this diagram is a very rough image, it should evolve over time.

(3) Interview with major stakeholders and solicit their needs.

(4) Select and define the collected needs.

(5) List major functional requirements.

(6) List Actors.

(7) Analyze Overall Risks.

(8) List major quality requirements for each quality characteristic such as safety, reliability and usability. A quality model, e.g. ISO/IEC9126-1 should be used for this purpose.

(9) List required constraints and conditions, including total budget, delivery date, hardware and communication network environment and available human resources.

(10) Refine system into sub-systems and re-define outline statement. Refine system overview diagram and project description based on the defined sub-systems.

Detailed Requirements Engineering
Iterate the process explained in clause 5.2 for each sub-system more accurately and precisely.

(1) State an outline of each sub-system in order to solicit detailed needs in terms of Quality-In-Use Requirements.

(2) Draw a Use Case Diagram for Each Sub-system.

(3) Interview the major stakeholders, especially the major users and solicit their needs.

(4) Identify and Clarify Context-Of-Use (COU).

COU may be started with a Use Case Scenario, but it must have more information, which especially relates to quality requirements and should be more formal. Use a tool that supports describing COU and that converts the COU to XML.

COU should include but is not limited to users and their profiles, other actors, target tasks, methods of usage,
environment, frequency of use, potential risk, time constraints etc. The following is an example of a COU description.

Actor:
Name: user
ID: Sub-system A, User type 1
Profile:
Age range: (~10, 11~20, 21~50.)
Sex: (Male, Female, Both)
Experience: None
Skill of operation: None

Task:
Name: Order entry
ID: SSO-1
Description: A user selects a product, type in quantity, select payment method.....

Constraint:

Environment:
Hardware:
Operating System:
Communication network:

(5) Analyze the Context-Of-Use and specify functional requirements for each Use Case.
(6) Select actors from COU and categorize Typical Users
(7) Analyze the Risks for Each Use Case and reflect them in the quality requirements
(8) Specify Quality-In-Use Requirements for each Use Case

The following is an example of a QIU specification.

Quality-In-Use Requirements
Use case ID:
QIU characteristic:
Effectiveness requirements:
Safety Requirements:

(9) Specify External Quality Requirements for each quality characteristic such as reliability and usability.

External Quality Requirements
Functionality:

Security:
Interoperability:
Reliability:
Usability:

(10) List required constraints and conditions, including total budget, delivery date, hardware and communication network environment and available human resources.
(11) Measure the QIU and Analyze the Results
(12) Refine the system outline statement, system overview diagram and project description.

6. ISSUES

Though experiences in an experimental scale at a university and companies with the method are promising, there are some issues for improvement. Examples include:

(1) Formalize the method: The method at this moment is not well defined. More effort is required for formalizing the method.

(2) Validation of the method by empirical studies: After the method is defined in a formal specification, the method should be distributed for beta test for the purpose of empirical validation.

(3) Methodologies from quality requirements to design: Some quality requirements should be reflected in the functional requirements. Some others may be reflected to the program architecture, structure and programming style. Guideline for this process should be developed.

(4) Support requirements change: Changes in requirements are always inevitable. At this stage of maturity, the method does not support requirements change. A method that predicts possibility of future changes and reflects it in the design should be developed.

(5) Quality measures: Requirements should be objective and quantitative. It also should provide evaluation criteria for delivery or acceptance. ISO/IEC JTC1/SC7 developed ISO/IEC 9124-2, -3 and -4 Quality metrics series technical report. JTC1/SC7 is also planning to revise the series for international standard as parts of SQuaRE series of international standard. However, in order to make it practical, more time and effort are needed [2] [3] [4] [5].

(Note: For the latest information, refer to http://www.jtc1-sc7.org/)

(6) Quality requirements standard: ISO/IEC JTC1/SC7 is developing a “Quality requirements” international standard as a part of SQuaRE series international standard. However, in order to make it practical, more time and effort are needed [2].
7. CONCLUSION

In this position paper, conceptual models for quality requirements are presented. Then definitions of needs, requirements and quality requirements are defined. Requirements for a method that support quality Requirements Engineering are also stated. Then a quality Requirements Engineering and specification methodology is provided with a simple example. While writing the paper, issues are identified. In order to find solutions for these issues, experts’ opinion and further discussion are important. For this respect, the workshop is a good opportunity. The author believes that international standard can contribute for improving software quality for critical systems. Unfortunately both approaches are time consuming. However, we cannot wait developing and using critical systems until we find the best solution. Meanwhile, we are obliged to use better solutions existing candidates.

ACKNOWLEDGEMENTS

The author express acknowledgement to ISO/IEC JTC1/SC7/WG6 members for their dedicated contributions for developing ISO/IEC 9126 series and SQuaRE series of international standards. I also thank to reviewers who read carefully and send me very informative comments.

REFERENCES

1. Azuma, M., QUALITY IN USE; Its Concept, Metrics and Methodology, Proceedings 2WCSQ, 2000