A Literature Survey on standards for software product quality

Shreyas Lakhe
B.E. 3rd Year student
College of Engineering, Pune
Nagpur. 440010 (India)

R. R. Lakhe
Director
Shreyas Quality Management System
Nagpur. 440010 (India)
rlakhe_cha@sancharnet.in

Rupali R. Dorwe
Priyadarshini College of Engineering
Nagpur, India
Rupalidorwe25@gmail.com

Ashwin B. Ganorkar
Priyadarshini Bhagwati College of Engineering
Nagpur, India
ash_ganorkar16@yahoo.com

***Abstract*— In this presentation, overview of current standards for software product quality are introduced followed by relationship between the standards. Then the concept and, Quality Requirements, usability and the model are explained.**

INTRODUCTION

The business value of a software product results from its quality as perceived by both acquirers and end-users. Therefore, quality is more and more often seen as a critical attribute of the software product, since its absence results in dissatisfied users and financial loss, and may even endanger lives. Increasing recognition of the importance of software quality causes a shift in the "center of gravity" of software engineering from creating a technology-centered solution toward satisfying the stakeholders. Software development organizations confronted with such a shift are, in general, not adequately equipped to deal with it.

There are a plethora of standards available for systems engineering, published by the International Organization for Standardization (ISO), the International Electrotechnical Commission (IEC), and the Institute of Electrical and Electronics Engineers (IEEE), Many of the standards cross-reference each other, and some are specific to a domain, e.g. software vs. systems. Furthermore, as technology changes, some of the standards become obsolete or outdated. For example, ISO/IEC Technical Report 24766 is a nice guideline for evaluating requirements engineering tools, yet it does not consider support for product lines (e.g. capture of variation points).

There are two international standard series, i.e. ISO/IEC 9126 and 14598, which are closely related to each other. ISO/IEC 9126 - Software product quality [1] and ISO/IEC 14598 - Evaluation of software products [2]. Table 1 presents the unit standards included in these two series.

No.	Title
9126- 1	Software Engineering –Product quality –Quality model
9126- 2	Software Engineering - Product quality – External quality metrics
9126- 3	Software Engineering - Product quality –Internal quality metrics
9126- 4	Software Engineering - Product quality –Quality in use metrics
14598-1	Information Technology –Software product evaluation –General overview
14598-2	Information Technology –Software product evaluation –Planning and management
14598-3	Information Technology –Software product evaluation –Process for developer s
14598-4	Information Technology –Software product evaluation –Process for Acquirers
14598-5	Information Technology –Software product evaluation –Process for Evaluators
14598-6	Information Technology –Software product evaluation – Documentation of evaluation modules
15288	Information Technology -- Life Cycle Management --System Life Cycle Processes
15939	Information Engineering -- Software Measurement Process

Table 1. List of ISO/IEC standards referenced

In parallel, other ISO working groups developed other process related standards for the engineering of software, and in particular ISO 15288 on life cycle phases, and ISO 1 5939 on the software measurement process (Table 1). The first generation of these product -related and process related standards are currently in their final ISO publication stage but, having been developed independently, their usage by practitioners will be particularly challenging. While ISO software experts are already at work defining strategies to develop the next generation of these standards, help is needed

by practitioners to understand, deploy and leverage ISO standards that are now becoming avail able to them.

RELATIONSHIP BETWEEN STANDARDS

The standards described here often contain project execution requirements (i.e. requirements governing how a product is built) in projects with legal authorities, showing up in public requests for proposals. These often end up in contracts, and the systems engineer then must be able to show that the development process and the specifications produced are compliant with the requested standards. So a certain familiarity with these standards is of benefit to the engineer, and more so if the engineer has plans to become a certified requirements engineer (e.g. through the International Council on Systems Engineering (INCOSE) or the International Requirements Engineering Board (IREB)). With ISO, IEC, and IEEE harmonizing their standards and due to the complexity of the domain, it is hard not to get confused, with the various relationships between standards.

Florian Schneider and Brian Berenbach has chosen standards that define the most important terms of requirements engineering, impose requirements on the engineering process, the tools used therein and the artifacts created with the tools in the process. As they are relatively new and thus might not be known by many, they also added two standards regarding quality requirements. The relationships between the chosen standards are shown in Figure 1.

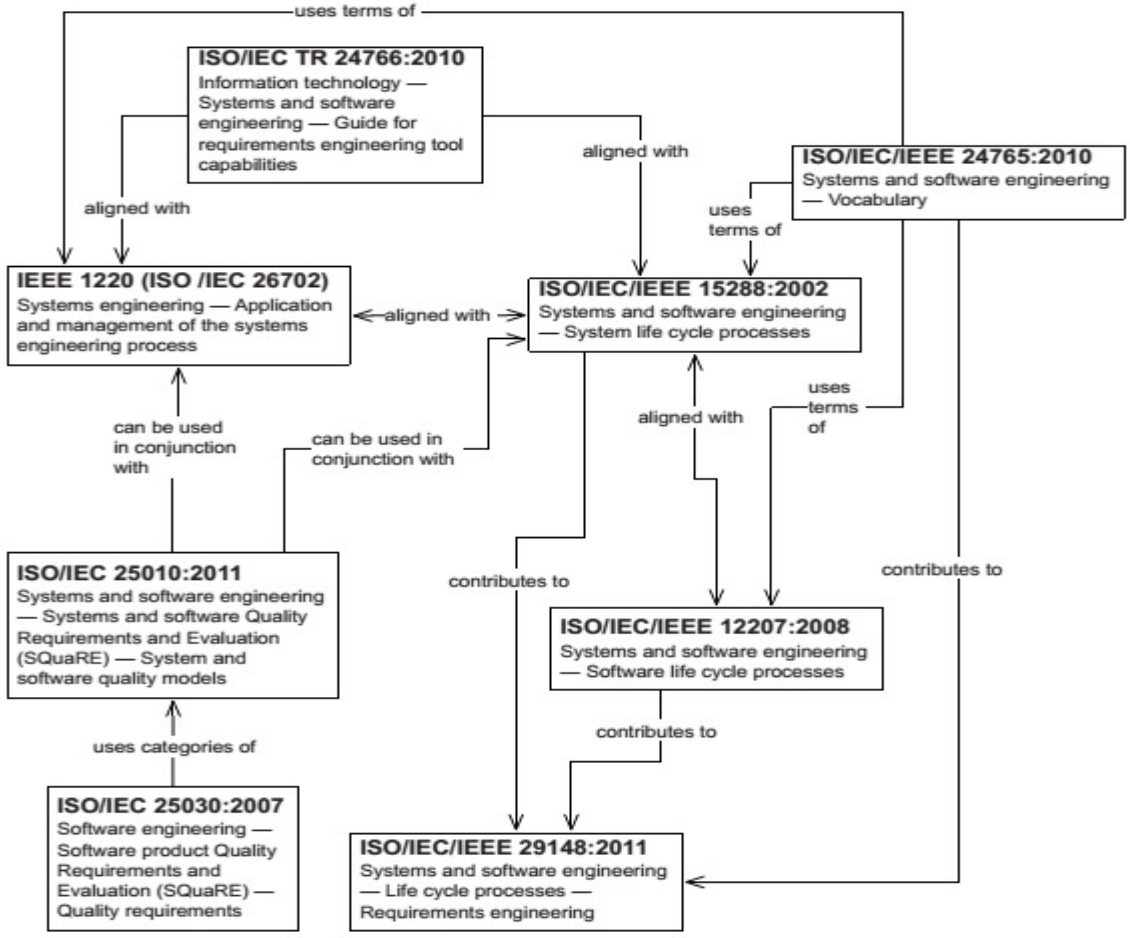

Figure 1: Relationships of the standards

There are two international standard series, i.e. ISO/IEC 9126 and 14598, which are closely related to each other. It is shown in figure

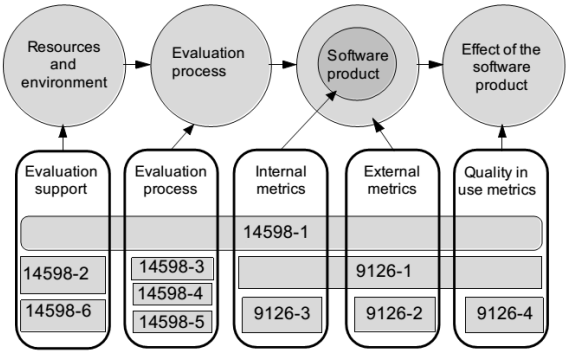

Figure 2: Current architecture of ISO/IEC 9126 and 14598 series

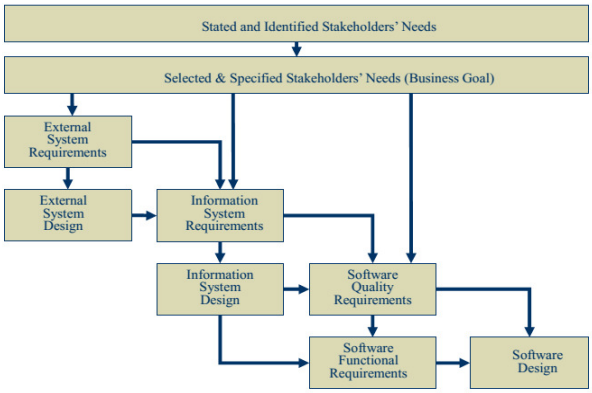

Figure 3:Need and requirements of system and software

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.

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. The totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs

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.

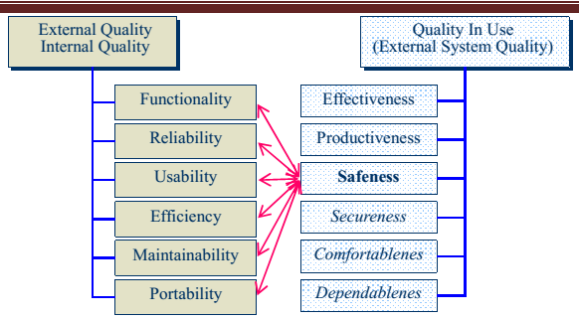

Figure 4: Modified ISO/IEC 1923-1 Quality Model

Internal Quality Requirements specify the level of required 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 Requirementsare 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

Probably, apart from Functional Requirements, Reliability Requirements is the most popular concept and frequently specified. It can be specified qualitatively using such measures as MTBF (Mean Time Between Failure) and MTTR (Mean Time To Repair). Software reliability is influential to safety of the External-System.

Usability Requirements should be stated by considering the users' profile, such as experience, operational skill, eyesight, and taste. In the case of defining Usability Requirements for critical systems in which misjudgments and/or improper operation by a user may cause a serious disaster, special care should be taken for specifying Usability Requirements.

Quality evaluation division
This division consists of the following four parts.

Evaluation process overview(revision of 14598- 1)

This part succeeds 14598-1, but inherits only general part for product evaluation. It introduces the other parts of this division . It contains general requirements for specification and evaluation of software quality and clarifies the general concepts. This part provides a framework for evaluating the quality of all types of software product and states the requirements for methods of software product measurement and evaluation.

Developers process(revision of 14598-3)
This part provides requirements and recommendations for the practical implementation of software product evaluation when the evaluation is conducted in parallel with the development and carried out by the developer.

Acquirers process(revision of 14598- 4)
This part contains requirements, recommendations and guidelines for the systematic measurement, assessment and evaluation of software product quality during acquisition of "off-the -shelf" software products, custom software products, or modifications to existing software products.

Evaluators process(revision of 14598-5)
This part provides requirements and recommendations for the practical implementation of software product evaluation, when several parties need to understand, accept and trust evaluation results.In particular, it may be used to apply the concepts described in ISO/IEC 9126-1n.

USABILITY
The term usability was coined some 10 years ago in order to replace the term "user friendly "which by the early 1980s had acquired a host of undesirably vague and subjective connotations .However, in the intervening years, the word usability itself has become almost as devalued as the term it was intended to supplant. There are still many different approaches to making a product usable, and no accepted definition of the term usability. The definitions which have been used derive from a number of views of what usability is. Three of the views relate to how usability should be measured:

• the product-oriented view, that usability can be measured in terms of the ergonomic attributes of the product;

• the user-oriented view, that usability can be measured in terms of the mental effort and attitude of the user;

• the user performance view, that usability can be measured by examining how the user interacts with the product, with particular emphasis on either- ease-of-use: how easy the product is to use, or- acceptability: whether the product will be used in the real world.

These views are complemented by the contextually-oriented view, that usability of a product is a function of the particular user or class of users being studied, the task they perform, and environment in which they work.

For example, the definition given in the ISO standard for software qualities (ISO 1991b) is product and user-oriented:

"a set of attributes of software which bear on the effort needed for use and on the individual assessment of such use ..."

The proposed ISO ergonomics definition (Brooke et al 1990) is usage, user and contextually oriented:

"the effectiveness, efficiency and satisfaction with which specified users can achieve specified goals in a particular environment".

Eason's (1988) definition is ease-of-use oriented:

"the degree to which users are able to use the system with the skills, knowledge, stereotypes and experience they can bring to bear".

The position taken by the ESPRIT MUSiC project is that a complete definition of usability must encompass all these views. Usability is a function of the ease of use (including learn ability when relevant) and the acceptability of the product and will determine the actual usage by a particular user for a particular task in a particular context. The current MUSiC definition of usability is: the ease of use and acceptability of a system or product for a particular class of users carrying out specific tasks in a specific environment; where 'ease of use' affects user performance and satisfaction, and 'acceptability' affects whether or not the product is used.

Ease of use determines whether a product can be used, and acceptability whether it will be used, and how it will be used. Ease of use in a particular context is determined by the product attributes, and is measured by user performance and satisfaction. The context consists of the user, task and physical and social environment. The relationship between these factors is shown in Figure 5.

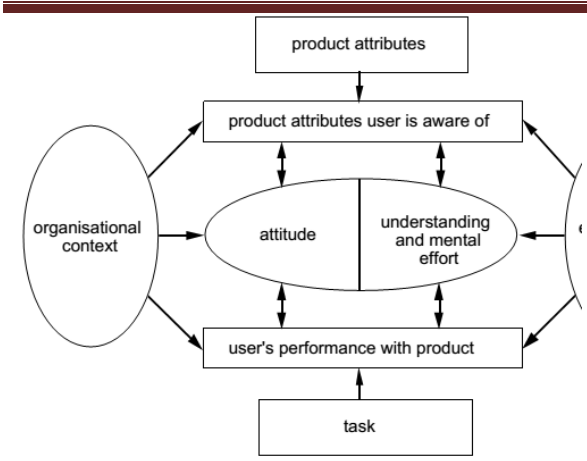

Figure 5: Determinants of usability

The product attributes which contribute to usability include the style and properties of the interface, the dialogue structure, the nature of the functionality, and any other relevant properties such as system efficiency and reliability. Measures of attitude and performance provide the criteria which determine whether the design of the attributes is successful in achieving usability. In the future, analytical techniques may be able to predict attitude and performance from these attributes (Bösser 1991).

The distinction between product attributes and user performance leads to two very different approaches: either emphasis on the specification, design and evaluation of product attributes which determine usability, or concern with the specification and subsequent evaluation of criteria for the user's attitude and performance.

CONCLUSION

The work for developing standards has been just initiated. The different quality standards and their relations summarized in this paper, is the initial input to the project. The quality standards may be modified and should be continuously improved because, both technologies for the needs and the target software are changing very rapidly, and usually, the work for developing standard is time consuming because of the necessary formal balloting procedure.

REFERENCES

[1] ISO/IEC 9126: Software Engineering - Product quality. Part 1: 2001 –Parts 2 to 4: to be published in 2003, International Organization for Standardization, Geneva.

[2] ISO/IEC 14598: Information Technology –Software product evaluation. Parts 1 to 6: 1999 -2001, International Organization for Standardization, Geneva, 1999 – 2001

[3] W. Suryn, A. Abran, and A. April, “ISO/IEC SQuaRE. The second generation of standards for software product quality” Software Engineering and Applications, November 3 – 5, 2003, arina del Rey, USA

[4] Florian Schneider, Brian Berenbach “A Literature Survey on International Standards for Systems Requirements Engineering”, Procedia Computer Science vol. 16, pp. 796-805, 2013

[5] Motoei AZUMA “SquaRE The next generation of the ISO/IEC 9126 and 14598 international standards series on software product quality” European Software Control and Metrics Conference, London, England (2001)

[6] Azuma, Motoei. "Applying iso/iec 9126-1 quality model to quality requirements engineering on critical software." Proceedings of the 3rd IEEE Int. Workshop on Requirements for High Assurance Systems (RHAS). 2004.

[7] Suryn, Witold, et al. "Software product quality practices-quality measurement and evaluation using TL9000 and ISO/IEC 9126." Software Technology and Engineering Practice, 2002. STEP 2002. Proceedings. 10th International Workshop on. IEEE, 2002.

[8] Isi Castillo, Francisca Losavio, Alfredo Matteo, Jorgen Boegh. “REquirements, Aspects and Software Quality: the REASQ model”, Journal of Object Technology, vol. 9, no. 4, 2010, pages 69–91.

[9] Yiannis Kanellopoulos, Panos Antonellis “Code Quality Evaluation Methodology Using The ISO/IEC 9126 Standard”, International Journal of Software Engineering & Applications (IJSEA), Vol.1, No.3, July 2010