The Use and Usefulness of the ISO/IEC 9126 Quality Standard

Hiyam Al-Kilidar

Computer Science and Engineering,
University of New South Wales /
National ICT Australia
hiyama@cse.unsw.edu.au

Karl Cox

Computer Science and Engineering,
University of New South Wales /
National ICT Australia
karl.cox@nicta.com.au

Barbara Kitchenham

National ICT Australia
Sydney
barbara.kitchenham@nicta.com.au

Abstract

This paper reports an evaluation the utility of ISO/IEC 9126. ISO/IEC 9126 is an international standard intended to ensure the quality of all software-intensive products including safety-critical systems where lives will be at risk if software components fail. Our evaluation exercise arose from an experiment that required a quality assessment of outputs of the design process. Although ISO/IEC 9126 is intended to support evaluation of intermediate software products, both the experimental subjects (158 final year computer science and engineering student) and experimenters found the standard was ambiguous in meaning, incomplete with respect to quality characteristics and overlapping with respect to measured properties. We conclude that ISO/IEC 9126 is not suitable for measuring design quality of software products. This casts serious doubts as to the validity of the standard as a whole.

1. Introduction

Software quality is fundamental to software product success [1]. Yet quality as a concept is difficult to define, describe and understand [2]. Quality has a strong subjective element. For example, a factor one person identifies as indicating good quality (e.g. interface simplicity and elegance), another person may regard as an indicator of poor quality (e.g. lack of help for novice users). Examination of quality definitions, meanings and views in [1] describes quality as hard to define and measure but easy to recognize. However, quality experts including some with a software background have proposed models e.g. [3-5] not to measure quality itself but to measure surrogate attributes such that when combined can provide some notion of the quality of the product.

Many definitions have been introduced to define quality [1-14]. The International Standards Organisation (ISO) defines quality as: *"the totality of features and characteristics of a product or service that bear on its ability to satisfy specified or implied needs"*[6]. The IEEE defines quality as "*the degree to which a system, component, or process meets specified requirements and customer or user needs or expectations"* [15]. Essentially, both definitions are focused on satisfying the customer's need for the software product. Wong [14] listed no less than 15 different quality definitions and views of quality models. This shows that there is no one encompassing definition or view of quality. The quality view taken in any given situation depends upon the context, the meaning assigned to the quality attributes and the relationships between those attributes within that context.

The idea of quality is not new. Two decades ago, Garvin [16] identified a holistic view of quality in general that comprised of five different perspectives: transcendental, user-based, product-based, manufacturing, and value-based perspectives. Kitchenham and Walker [17] applied Garvin's view of quality to software. The focus of this paper is the product-view which assumes that quality is dependent on the inherent characteristics of a product and can be quantified based on the presence or absence of a number of attributes. The idea behind this research is that the ability to meaningfully quantify and measure the quality attributes of a software design will be useful in ensuring a higher quality software product because the design is a subset/component of the final product.

ISO and the International Electrical technical Commission (IEC) have developed the ISO/IEC 9126 Standards for Software Engineering – Product Quality [18-21] to provide a comprehensive specification and evaluation model for the quality of software products. In this paper we report on usage and application of ISO/IEC 9216. When we began our research we decided to use ISO/IEC 9126 to assist with measuring the quality of software designs. There were criticisms of the first version of ISO/IEC 9126 suggesting it was not comprehensive [11, 22], was difficult to understand [11, 13], and arbitrary with respect to the selection of characteristics and sub characteristics some of which were

0-7803-9508-5/05/$20.00 ©2005 IEEE

unverified and perhaps unverifiable [23]. However, the standard was substantially revised in 2001 and 2002, so we felt it was important to use an approved international standard as our starting point. Organisations may be persuaded to adopt ISO/IEC 9126 because it is an international standard addressing an extremely important concern i.e. the quality of software. It is therefore important that use of ISO/IEC 9126 be properly tested and evaluated. Furthermore, it is surprising that, unlike SPICE, ISO/IEC 9126 has not been subject to a formal evaluation as part of its production process.

Our interest in ISO/IEC 9126 arose because one of the authors was undertaking an experiment on pair-design. The experiment required that subjects either work alone or as pairs to produce a software design. The experiment required that the subjects be given a checklist to help them produce a good quality design, and that the designs produced by the subjects be evaluated for quality. Initially, the experimenter planned to use ISO/IEC 9126 for both purposes. This paper reports the difficulties encountered trying to use ISO/IEC 9126 for these purposes.

The paper takes the following format. In section 2, we discuss ISO/IEC 9126. In section 3, we briefly describe the experiment conducted to provide some context to the findings discussed in this paper. In section 4 we present a detailed discussion of our experiences of the use and application of ISO/IEC 9126. In section 5 we provide some conclusions.

2. ISO/IEC 9126

The ISO/IEC 9126 standard is divided into four parts [18-21].

- Part 1. Software Engineering–Product quality "quality model": describes the quality model framework explaining the relationships between the different approaches to quality as well as identifying the quality characteristics and sub-characteristics of software products.
- Part 2. Software Engineering–Product quality–External metrics: describes the external metrics used to measure the characteristics and sub-characteristics identified in part 1.
- Part 3. Software Engineering–Product quality–Internal metrics: describe the internal metrics used to measure the characteristics and sub-characteristics identified in part 1.
- Part 4. Software Engineering–Product quality–Quality in use metrics: identifies the metrics used to measure the effects of the combined quality characteristics for the user.

The first three parts are concerned with describing and measuring the quality of the software product, the fourth part evaluates the product from the user point of view.

Since the ISO/IEC 9126 standard consists of four substantial documents, we only provide a brief overview due to paper length restrictions. ISO/IEC 9126 is intended to:

- Provide a comprehensive specification and evaluation model for software product quality.
- Explicitly address user needs of a product by allowing for a common language for specifying user requirements that is understandable by users, developers and evaluators.
- Objectively evaluate quality of software products based on observation not opinion.
- Make quality evaluation reproducible [24].

The quality model proposed in ISO/IEC 9126 (see Fig. 1) essentially consists of two main parts: 1) internal and external quality attributes and 2) quality in use attributes.

The first part identifies the quality of a software product through six quality characteristics, Namely: Functionality, Reliability, Usability, Efficiency, Maintainability an Portability. Each characteristic is subdivided into related sub-characteristics. Each sub-characteristic is further described by appropriate external and internal quality attributes that can be measured by specified metrics. The metrics listed in [19, 20] are described in terms of purpose, method of application, measurement, formula, data element computations, value interpretation, scale and measure type as well as source of input and target audience. The internal attributes of a product form the basis for achieving required external quality performance which, in turn, forms the basis for achieving quality in use.

The second part, 'quality in use' specifies four quality characteristics namely, effectiveness, productivity, safety and satisfaction, which ISO/IEC 9126 suggests are indicative of the user's view of quality based on the combined effect of the attributes specified in Part 1 of the standard [21]. This necessitates identification of the users' quality requirements to specify the external and internal quality attributes of the product, and hence, the quality characteristics, sub-characteristics and related metrics.

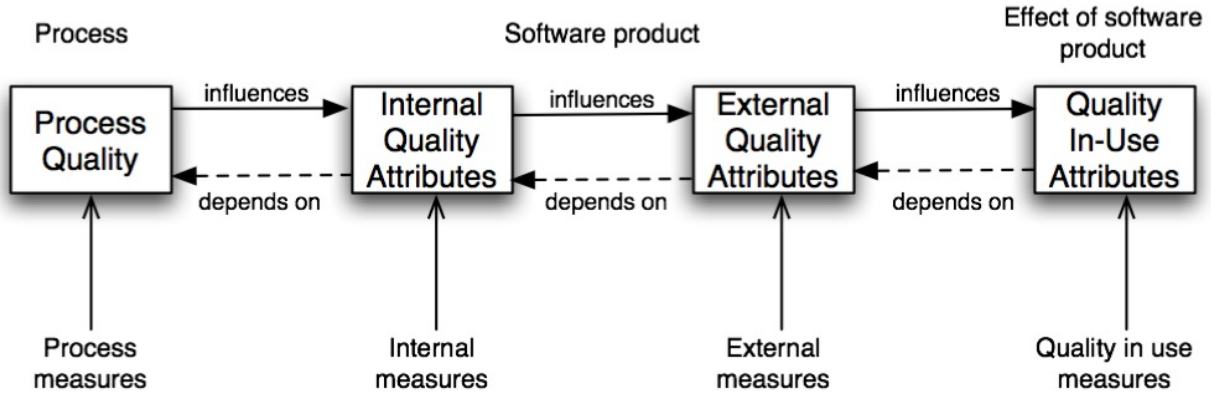

Fig. 1. Quality Model Framework Lifecycle of ISO/IEC 9126 (adapted from [18])

ISO/IEC 9126 states that within a predefined context, the quality of a software product *"can be evaluated by measuring internal attributes (typically static measures of intermediate products), or by measuring external attributes (typically by measuring the behaviour of the code when executed), or by measuring quality in use attributes* ...[where] *appropriate internal attributes of the software are a pre requisite for achieving the required external behaviour and the appropriate external behaviour is a pre-requisite for achieving quality in use"* [18]. The ISO/IEC 9126 standards also suggest that in the early stages of the development process, available intermediate products can be evaluated using the internal metrics as detailed in [18, 19].

3. Overview of Experiment

In this section we provide a brief overview of an experiment as reported in [25, 26]. The outcome measure we planned to use in the experiment was the quality of intermediate design products. The experimental subjects were instructed to produce design documents consisting of DFDs, RDBs and functional Interface Diagrams IDs for a web based application. Attributes and metrics derived from ISO/IEC 9126 were used to evaluate the quality of the design artefacts.

The aims of the experiment included investigating:

1. Whether it is possible and meaningful to examine the intermediate design products for quality
2. Whether the characteristics, sub characteristics and metrics as described in ISO/IEC 9126 standard of Internal Metrics [19] are applicable to the design products as suggested by the standard.
3. How the ISO/IEC 9126 quality attributes can be incorporated, measured and evaluated for design products.

The experiment ran as part of the course work for Software Project Management in the School of Computer Science and Engineering at the University of New South Wales. The 158 participants were in their final year and were asked to produce design outputs as well as plan and track their design activities using Microsoft Project. All subjects were asked to design two related modules of a realistic network project management tool that is capable of effectively and efficiently managing a large number of autonomous collaborating partners.

3.1 Inputs to experiment

Subjects were given a Software Requirements Specification (SRS) that detailed information about the system context [25] together with a set of conceptual data models in ERD format, and a limited set of use cases. The ERDs were used to identify principle entities for which the attributes must be managed in the project management database and the principle relationships between those entities that must also be managed in the database. The use cases described the scope of the system's behaviour to be used for the functional design [27] and provided enough information to show system behaviour without describing the system interface design [25].

In addition, the subjects were provided with a unified set of quality attributes and metrics derived by the experimenters from ISO/IEC 9126 to be used as guidelines and checklist of quality measures.

3.2 Outputs of experiment

The subjects were asked to produce:

1) Process model in DFD format, which provided a representation of the data processing structure and logic of the flow of data in the system. The DFDs provide a network representation of the structure and logic of the system partitioning into its components according to the use cases. The subjects were instructed to start with a context diagram, then proceed to create level 0 DFD, then to detail and expand the processes until 3rd level DFD according to the use cases, and provide pseudo code for the 3rd level diagram.
2) Relational database storage model in table format, that specified the attributes associated with each entity.
3) Interface model with screen shots illustrating the external functional behaviour only.

The design outputs would, theoretically, be handed to developers/programmers for implementation.

3.3 Practical Problems Using ISO/IEC 9126

Prior to taking part in the experiment, the students , who were to play the role of designers in the experiment, were engaged in an exercise designed to identify the quality attributes and metrics from the developers' point of view. In this exercise, students were presented with ISO/IEC 9126 and asked to identify the quality attributes they thought should be incorporated into a quantitative instrument to enable the measurement and evaluation of the quality of the design products. Our original intention was to summarise the students' ideas into a consolidated measurement instrument. However, this did not prove to be possible. Two significant problems arose:

1. The students found it difficult to interpret ISO/IEC 9126 because many terms were ambiguous.
2. Students found the concept of usability in ISO 9126 difficult to understand.

3.3.1 Ambiguity in ISO/IEC 9126

As suggested earlier, because of its subjective nature, quality can mean different things to different people. Therefore, adopting a unified terminology such as that of a standard can be considered a step in the right direction to remove ambiguity. Such a standard should provide a common language between stakeholders, e.g. users and their requirements, developers, evaluators and managers. One of the stated aims of ISO/IEC 9126 is to provide such a common language.

We found, however, that the 'common language' proposed by ISO/IEC 9126 was far from practical. It did not have a standard interpretation. Indeed, the standard does not provide a glossary of terms. This only makes the implementation of the standard more subjective and open to ambiguity. The experimental subjects found it difficult to interpret some of ISO/IEC 9126's characteristics and metrics for quality attributes because of their subjective interpretation of the terminology of ISO/IEC 9126.

3.3.2 Usability in ISO/IEC 9126

Usability is described in the ISO/IEC 9126 standard as, "*The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions*" [19]. Yet the document on internal metrics (part 3) of the standard [19], suggests usability sub characteristics and metrics such as *evident functions* can be identified as functions that are evident to the user. However, the metrics are identified and used by developers, not end users. A more typical way of interpreting usability, aside from its use in ISO/IEC 9126, would be that usability is not a common internal design factor, but it is in fact generally accepted as a property of the system in-use from the end-user perspective. We recognise that developers have to consider usability in design and that there are some wizards that help address relationships between components. However, we do not believe this is a standard interpretation of usability. Even when designers have to conform to GUI standards, such as not overcrowding dialogue boxes, use of text styles, colour, these are all about the external representation, i.e. the external design in terms of look and feel of the system from the user perspective. Most of the 158 subjects in the experiment responded in two ways to the description of usability given by ISO/IEC 9126.

1) I don't know what ISO/IEC 9126 means.
2) I *think* ISO/IEC 9126 means '…'

Although the subjects of the experiment were students, and some argue that students do not understand much without proper training, the experimenters *also* had problems in interpreting the standard. This implies that some training or guidelines for use of ISO/IEC 9126 is necessary. What could be even more problematic is that experienced designers will be able to construct a number of different interpretations of the standard, implying that ISO/IEC 9126 is not a *standard* at all.

4. Detailed Evaluation of ISO/IEC 9126

As a result of the problems the students encountered we reviewed ISO/IEC 9126 from the viewpoint of design artefacts in more detail. Our findings are presented below.

4.1 Ambiguous Metric Definitions

The Function of the internal metric *Functional Compliance*, is stated as "*How compliant is the functionality of the product to applicable regulations, standards and conventions*" [19]. The input to this measurement includes design documents which in our

case were DFDs. However, the method of application states, "*Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification,*" [19]. It is not clear whether ISO/IEC 9126 is referring to,

1. Compliance of design products (e.g. DFDs) to the regulations of a specific industry. Or,
2. Compliance of design products to the rules for producing these products (e.g. how to correctly draw a DFD). Or,
3. Compliance of design functionality(ies) to the functionality(ies) as they are stated in the software requirements specification.

Moreover, if the compliance refers to the interpretation in point 3 above, would this make the particular metric redundant with regards to other metrics in the standard? For example, the *Functional Adequacy* internal metric's 'method of application' states, "*Count the number of implemented functions that are suitable for performing the specified tasks… The following may be measured [in] all or parts of design specifications."* That is, does our third point in the above list mean the same as this description of *Functional Adequacy*? Does this mean we measure the same thing twice? Isn't *Functional Adequacy* redundant in this case? If point 3 is to be ruled out on the basis of redundancy with other metrics, should points 1 and 2 be combined or measured separately?

Another example of ambiguity is in the difference between the two metrics of *Functional Implementation Completeness* and *Functional Implementation Coverage* [19]. The method of application of *Functional Completeness* is to "*count the number of missing functions detected in evaluation and compare with the number of function described in the requirement specifications*". The method of application for *Functional Implementation Coverage* states, "*Count the number of incorrectly implemented or missing functions and compare with the number of functions described in the requirement specifications*" [19]. The description of the *Functional Implementation Completeness* implies that this metric is a subset of *the Functional Implementation Coverage* metric. If both metrics were to be considered, wouldn't that be redundancy? That is, completeness and coverage essentially mean the same thing. In the context of conceptual models, Lindland et.al [28] suggest that completeness and validity alone are sufficient to specify the quality properties needed to evaluate a conceptual model. We suggest the same is true of designs. In the context of a design, completeness means that each element in the requirements specification is represented in the design and validity means that each element in the design is correct.

Completeness in design would mean that the design includes all elements that are correct and relevant, and

Validity in design would mean that all design elements are correct and relevant to requirements.

4.2 Counts and Counting Measures

ISO/IEC 9126 relies on a counting technique to reduce subjectivity. However, as Einstein put it, "not everything that can be counted counts, and not everything that counts can be counted" [29]. It may be useful to incorporate other more conventional measures of design quality related to coupling and cohesion, for example [3], although this would require the provision of guidelines to *measures* not counts.

On another level, ISO/IEC 9126 is not clear about whether *ALL* inputs to measurement should be used together in conjunction or whether they should be used as appropriate or available? Indeed, ISO/IEC 9126 provides no guidance, heuristics, rules of thumb, or any other means to show how to trade off measures, how to weight measures or even how to simply collate them. It is left to the individual to decide how to deal with such issues.

4.3 Attributes and Counts Assigned to Inappropriate Lifecycle Phases

With regard to the accuracy and interoperability metrics, design documents are considered inputs to measurements in the *Computational Accuracy*, *Data Exchange* and *Interface Consistency* metrics [19]. Yet the method of application requires counts of *implemented* functions. The question that this poses is how can designers measure these metrics for intermediate products when nothing is yet implemented? There seems to be an assumption that design is the same as implementation.

These interpretations indicate that the language of ISO/IEC 9126 is inadequate. The fact that ISO/IEC 9126 is open to interpretation shows it is failing its purpose. For example, Usability is defined in part 3 of ISO/IEC 9126 as an external quality attribute with external metrics, which the standard claims can be measured during design. However, when we consider matter of usability, we usually think of the end-user view of the system. Does ISO/IEC 9126 consider the end-users of design to be the developers/programmers who will implement the design? If this is the case, it is not made explicit. Furthermore, in part 4 of the standard, which explicitly addresses quality in use from the user perspective, ISO/IEC 9126 does not address usability at all. Has usability been placed in Part 3 by mistake? If this is the case, the standardization and document production process is poor.

We found some characteristics and metrics difficult to operationalise as measures. For example, *Function Understandability* is described in 'method of application' of the ISO/IEC 9126 standard as "*count the number of user interface functions where purposes is understood by the user and compare with the number of user interface functions*". Not only is the description poor grammar but we also believe it will be difficult for the developer to

assume, or not, that the user will, or will not, be able to understand the purpose of all the interface functions. As an example, in a strategic or organisation-wide system, there will be many functions that users will know little or nothing about because these are targeted at many different groups of end-users right across the organisation, both vertically and horizontally.

If the "user" in the *Function Understandability* metric is the end user, how is the programmer meant to access the end-user? Most organisations have analysts and managers who talk to end users. Of course, if all software is produced using XP [30] then this interpretation of the metric makes sense since, ideally, there is always a customer/end-user representative available for the programming team. But since most of the world is not like this, a programmer will have access to customer/end-user needs by tracing backwards to specification and requirements documents. Recent research has confirmed that users and developers have distinctly different viewpoints. The developer/programmer view of project success has been demonstrated to be different to other project members and to be different to an end-user view of success [31, 32]. So for ISO/IEC 9126 to assume that a developer would be able to assess whether a user thinks a function is understandable or not is somewhat dubious.

4.4 Missing Attributes and Measures

ISO/IEC 9126 has a column called 'sources of input to measurement' that identifies functionality, usability and portability as being sourced from designs. We feel there is serious problem with ISO/IEC 9126 not recognizing characteristics such as maintainability and reliability also need to be sourced from designs; for example, fault tolerance features and modularity both need to be considered during design.

Our experience in using ISO/IEC 9216 suggests that some characteristics are too abstract. We needed to consider the nature of each entity being measured and this resulted in the addition of extra attributes. For example, we considered attributes such as validity and modularity as important measures of the quality of designs. ISO/IEC 9126 does not consider them at all.

ISO/IEC 9126 Part 4, Quality In-use metrics[21], includes characteristics such as effectiveness and productivity. These are more the concerns of developers rather than users and clients who would be more interested in issues such as ease of use, timely delivery and cost, for example.

5. Conclusions

This paper reports on the application of ISO/IEC 9126 to software design documents in an empirical study. The paper relates the experiences of using ISO/IEC 9126 to assess the quality of design documents that were the outputs of an experiment. Jackson [33] states that the generality of a method is inversely proportional to its utility. To paraphrase Jackson, we found the generality of ISO/IEC 9126 to be inversely proportional to its utility. ISO/IEC 9126 is hindered by its generality. Our experience of ISO/IEC 9126 shows that it is specific enough to be unclear or misleading yet is meant to be general enough to be applicable to all software products. Thus, it is not specific enough to be useful for any particular design product.

We summarise the problems with ISO/IEC 9126 for design quality as follows:

- Some concept definitions are ambiguous, e.g. *functional compliance*.
- Some concept definitions overlap, e.g. *functional implementation completeness* and *functional implementation coverage*.
- Overlapping definition of concepts can lead to multiple counting when metrics are constructed.
- The standard recognises reliability and maintainability as quality characteristics but does not refer to them when considering design products although most software engineers would agree that both characteristics need to be designed into products.
- The standard ignores other characteristics that might be important in design products such as validity and modularity.
- Simple Counts are insufficient to evaluate the quality of design.
- Some measures require information that is not available to the designers, such as *functional understandability*.
- Some measures require counting items that are not available from design documents, such as *computational accuracy* and *data exchange*.
- No guidelines or procedures are defined for accumulating the metrics into an overall evaluation.

In the light of its ambiguities and omissions, we conclude that ISO/IEC 9126 in its present format fails to achieve any of its stated objectives.

References

[1] B. Kitchenham and J. Walker, "A quantitative approach to monitoring software development," *Software Engineering Journal*, pp. 1-13, 1989.

[2] L. Bratthall and C. Wohlin, "Understanding Some Software Quality Aspects from Architecture and Design Models," presented at 8th IEEE International workshop on Program Comprehension (IWPC 2000), 2000.

[3] D. Budgen, *Software Design*. Essex, UK: Pearson Educational Limited, 2003.
[4] P. Crosby, *Quality is Free*. Maidenhead: McGraw Hill, 1978.
[5] D. Galin, *Software Quality Assurance, From theory to implementation*: Pearson education Ltd., 2004.
[6] ISO, "ISO 8402 Quality Vocabulary," in *International Organisation for Standardization*. Geneva, 1986.
[7] J. C. Munson, *Software engineering Measurement*: Auerbach Publications, 2003.
[8] S. L. Pfleeger, *Software Engineering Theory and Practice*: Prentice-Hall, Inc., 2001.
[9] R. S. Pressman, *Software Engineering, A Practitionar's Approach*, 2nd ed: McGraw-Hill Inc., 1987.
[10] I. Sommerville, *Software Engineering*, 4th ed: Addison-Wesley Publishers Ltd., 1992.
[11] I. Sommerville, *Software Engineering*, Sixth ed: Addison-Wesley, 2001.
[12] J. Tian, "Quality-Evaluation Models and Measurements," *IEEE Software*, pp. 84 -91, 2004.
[13] D. Wallace and L. Reeker, "Software Quality," in *Guide to the Software Engineering Body of Knowledge SWEBOK*, A. Abram and P. Bourque, Eds.: IEEE, 2001, pp. 165 - 184.
[14] B. Wong, "An Investigation of the Cognitive Structures used in Software Quality Evaluation; PhD Thesis," in *Information Systems, Technology and Management*. Sydney: University of New South Wales, 2002.
[15] IEEE, "IEEE Std 1074 -1997 - Standard for Software Life Cycle Processes," 1998.
[16] D. Garvin, "What Does "Product Quality" Really Mean?," *Sloan Management Review*, vol. 24, 1984.
[17] B. Kitchenham and J. Walker, "The Meaning of Quality," presented at Software Engineering, Southampton, England, 1986.
[18] ISO/IEC, "ISO/IEC 9126-1 Software engineering- Product quality- Part 1: Quality model," 2001.
[19] ISO/IEC, "ISO/IEC 9126-3 Software engineering -Product quality- part3: Internal metrics," 2002.
[20] ISO/IEC, "ISO/IEC 9126-2 Software engineering -Product quality- part2: External metrics," 2002.
[21] ISO/IEC, "ISO/IEC 9126-4 Software engineering -Product quality- part4: Quality In Use metrics," 2002.
[22] B. A. Kitchenham, S. G. Linkman, A. Pasquini, and V. Nanni, "The SQUID approach to defining a quality model," *Journal of Software Quality*, vol. 6, pp. 211-233, 1997.
[23] B. Kitchenham, "Software metrics. Measurement for software process improvement. (Chapter 4)," NCC Blackwell, 1996.
[24] T. Punter, R. Solingen, and J. Trienekens, "Software Product Evaluation," presented at 4th IT Evaluation Conference (EVIT-97), Netherlands, Delft, 1997.
[25] H. Al-Kilidar, R. Jeffery, A. Aurum, and C. Kutay, "Planning an Empirical Experiment To Evaluate The Effects Of Pair Work On The Design Phase Of The Software Lifecycle," University of New South Wales, Sydney 2003.
[26] H. Al-Kilidar, R. Jeffery, and A. Aurum, "Description of an Empirical Experiment to Measure effects of Pair work on the Design Phase," presented at International Association of Science and Technology for Development (IASTED)- Software Engineering, Innsbruck, Austria, 2004.
[27] A. Cockburn, *Writing effective Use Cases*: Pearson Education Corporate Sales Division, 2001.
[28] O. V. Lindland, G. Sindre, and A. Sølvberg, "Understanding Quality in Conceptual Modeling," *IEEE Software*, pp. 42-49, 1994.
[29] A. Einstein, "http://www.quotationspage.com/quotes/Albert_Einstein/."
[30] K. Beck, *eXtreme Programming explained, Embrace Change*: Addison-Wesley, 2000.
[31] J. M. Verner and W. M. Evanco, "In-house Software Development: What Software Project Management Practices Lead to Success?," *IEEE Software*, vol. 22, pp. 86-93, 2005.
[32] J. M. Verner, K. Cox, S. Bleistein, and N. Cerpa, "Requirements Engineering and Software Project Success: An Industrial Survey in Australia and the US," *Australian Journal of Information Systems (in press)*, 2005.
[33] M. A. Jackson, *Software Requirements and Specifications*: Addison Wesley, 1995.