Evolving a Corporate Software Quality Assessment Exercise: A Migration Path to ISO/IEC 9126

MARC-ALEXIS CÔTÉ
École de technologie supérieure, Montréal, Canada

WITOLD SURYN
École de technologie supérieure, Montréal, Canada

ROBERT A. MARTIN
MITRE Corporation

CLAUDE Y. LAPORTE
École de technologie supérieure, Montréal, Canada

This article examines the different ways in which the quality model behind MITRE Corporation's Software Quality Assessment Exercise (SQAE) can be migrated to ISO/IEC 9126:2001. The reasons why such a migration is desirable are detailed through a comparison of the quality models supporting ISO/IEC 9126 and SQAE. The problem is solved through the definition and usage of an abstraction layer between the two quality models. The resulting model is examined and further improvements are suggested to ensure compliance with ISO/IEC 9126.

Key words: assessment, measurement, quality factors, quality models, risk management, standards

SQP References

A Process Model of Software Quality Assurance/Software Quality Engineering
vol. 2, issue 4
Rodger Drabick

Applying Quantitative Methods to Software Maintenance
vol. 3, issue 1
Ed Weller

The Impact of National Cultural Factors on the Effectiveness of Process Improvement Methods: The Third Dimension
vol. 4, issue 4
Miklos Biro, Richard Messnarz, and Alfred G. Davison

INTRODUCTION

In *Quality Is Free*, Philip Crosby (1979) argued that it is not quality that is expensive, but rather the lack of it. The absence of quality thus poses a risk to the development of software. The sooner such risks are identified in the life cycle, the sooner appropriate corrective actions can be taken to improve the chances of a successful software development.

Over the last decade, MITRE Corporation has developed a software quality assessment exercise (SQAE) that can be used to quantify the risks associated with software. This tool has proved useful in helping a number of development teams analyze software risks; is freely available from MITRE (2004); has proved useful in assessing a variety of systems, architectures, and languages; and is being studied by the Software Engineering Institute (Williams and Bentrem 2004), which, along with Ecole de Technologie Superieure, has a no-cost license to use and study it. However, the quality model on which SQAE is based, the very definition against which quality is measured, is not based on an internationally recognized standard. This is not a flaw per se. However, because quality has proved to be such an elusive concept (Schulmeyer and McManus 1998), it is important for any tool measuring it have a solid foundation.

This article demonstrates approaches for how the SQAE can be migrated to take full advantage of the internationally recognized quality model defined by ISO/IEC 9126:2001 (ISO/IEC 2001a). This standard has recently undergone a major revision that makes it a likely candidate to be the solid foundation needed by SQAE to continue its evolution. ISO/IEC 9126 and SQAE will both be briefly introduced. These descriptions will be followed by a description of how the SQAE can conform to ISO/IEC 9126.

ISO/IEC 9126

What It Is

In 1991, the International Organization for Standardization (ISO) introduced ISO/IEC 9126 (1991): Software product evaluation—Quality characteristics and guidelines for their use. This standard aimed to define a quality model for software and a set of guidelines for measuring the characteristics associated with it. ISO/IEC 9126 quickly gained notoriety with information technology (IT) specialists in Europe as the best way to interpret and measure quality (Bazzana, Andersen, and Jokela 1993). However, Pfleeger (2001) reports some important problems associated with the first release of ISO/IEC 9126:

- There are no guidelines on how to provide an overall assessment of quality.
- There are no indications on how to perform the measurements of the quality characteristics.
- Rather than focusing on the user view of software, the model's characteristics reflect a developer view of software.

In order to address these concerns, an ISO committee began working on a revision of the standard. The results of this effort are the introduction of a revised version of ISO/IEC 9126 focusing on the quality model, and a new standard, ISO/IEC 14598 (ISO/IEC 1999), focusing on software product evaluation. ISO/IEC 14598 addresses Pfleeger's first concern, while the revision to ISO/IEC 9126 aims to resolve the second and third issues. ISO/IEC 9126 is now a four-part standard:

- ISO/IEC 9126-1 defines an updated quality model
- ISO/IEC 9126-2 defines a set of external metrics
- ISO/IEC 9126-3 defines a set of internal metrics
- ISO/IEC 9126-4 defines a set of quality-in-use metrics

Why It Is Important

ISO/IEC 9126 is the international standard for software quality that has been agreed upon by a majority of the international community and upon which some countries, such as Japan, have decided to standardize. It defines a common language relating to software product quality and is widely recognized as such, at least in Europe, where a survey indicates that it is known by at least 70 percent of the IT community (Bazzana, Andersen, and Jokela 1993).

FIGURE 1 Relationships between the different aspects of quality (ISO/IEC 2001a)

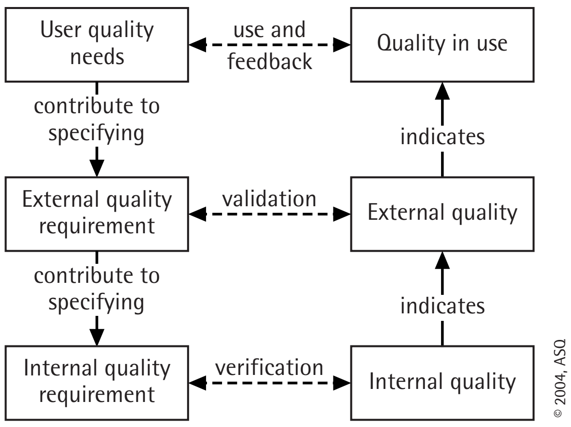
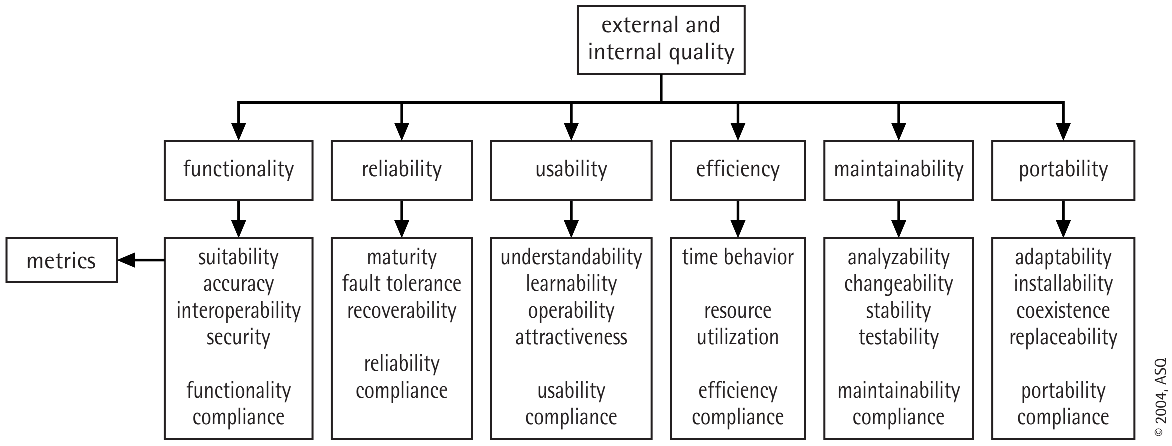


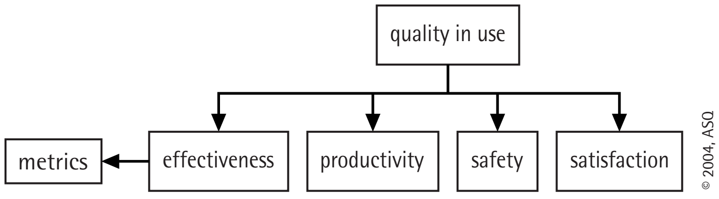
FIGURE 2 Three-layer model for internal and external quality (ISO/IEC 2001a)


external and internal quality
functionality
reliability
usability
efficiency
maintainability
portability
metrics
suitability accuracy interoperability security functionality compliance
maturity fault tolerance recoverability reliability compliance
understandability learnability operability attractiveness usability compliance
time behavior resource utilization efficiency compliance
analyzability changeability stability testability maintainability compliance
adaptability installability coexistence replaceability portability compliance
© 2004, ASQ

How It Works

ISO/IEC 9126 defines one quality model with three aspects: one for internal quality, one for external quality, and a final one for quality in use. The relationships among these three aspects of quality and their significance are illustrated by Figure 1.

FIGURE 3 Two-layer model for quality in use



The aspects for internal and external quality are quite similar. They both rely on a three-layer model composed of characteristics, subcharacteristics, and metrics (see Figure 2). Of course, the associated metrics are different for internal and external quality. The major difference with the first incarnation of ISO/IEC 9126 is the inclusion of suggested metrics (ISO/IEC 2003a; ISO/IEC 2003b) for measuring each subcharacteristic. It is important to note that these metrics are not normative (that is, a custom set of metrics can be defined, as long as they conform to annex A of ISO/IEC 9126-1).

Another important addition is a quality-in-use aspect (ISO/IEC 2001a). This part of the model aims at defining the quality attributes that are important for the end user and, therefore, addresses Pfleeger's third concern about ISO/IEC 9126. The quality-in-use aspect is illustrated in Figure 3.

As is the case with the internal and external quality parts, a set of informative metrics is associated with each quality-in-use characteristic (ISO/IEC 2001b). This model is very appropriate for giving an appreciation of the quality to end-user software.

SQAE

What It Is

Software maintenance can account for more than 60 percent of all effort expended by a development organization (Pressman 2001; Manna 1993). This is partly due to the fact that much of the software people depend on today is more than 15 years old and had to be migrated to different hardware platforms (Osbourne and Chikofsky 1990). However, most software organizations have traditionally focused on present risks rather than future (and more expensive) risks (Martin 2003). Short-term risks that are usually the immediate focus of a development or maintenance team include:

- Managing the initial development schedule
- Containing the development costs
- Providing desired functionality

This often results in software that is hard to maintain and entails unforeseen long-term costs. However, as software suppliers come and go, IT organizations must make management choices now, choices that they will have to live with for the next 15 years. How are those organizations supposed to assess the risk associated with such an important choice?

In order to provide a satisfying answer to this question, MITRE has created an SQAE providing a set of tools and evaluation methods that give a repeatable and consistent measure of the quality of the software and its associated risks (Martin and Shaffer 1996). The assessment of the quality provided by SQAE focuses on the risk associated with different quality areas and produces a list of risk drivers and mitigating elements that can help software developers and managers reorient their development effort and assist IT organizations in making judicious choices when selecting a software developer and/or maintainer. The SQAE is primarily intended for third-party evaluations, where an independent group is assessing and evaluating the quality of the software products being developed. By design, the SQAE is very rapid, economical, and the results are independent of the individuals involved in any particular assessment.

The quality model behind the SQAE method is based on the earlier work of Boehm (1978), McCall, Richards, and Walters (1997), and Dromey (1995) and not on the internationally recognized quality model proposed by ISO/IEC 9126, since the two efforts (SQAE and ISO/IEC 9126) were developed in parallel.

Why It Is Important

SQAE has been used to analyze more than 100 systems. This represents more than 50 million lines of code written in a large number of programming languages, from assembler to 4GL. It has also been used to assess the quality of systems ranging from 4000 lines of code, to more than 6 million lines of code.

The SQAE incorporates rigid scoring criteria that allow it to incorporate and constrain the subjective nature of some of its nonmetric evaluation criteria, and it integrates the analysts' observations and annotations into the findings rather than using a simple numerical score to create the total risk profile for the evaluated system. SQAE has proved through time that it can provide useful assessments of the quality risks of a variety of software packages in a repeatable and assessor independent way (Martin 2003; Martin and Shaffer 1996). The Department of Defense and other U.S. government agencies, as well as Ecole de Technologie Superieure, have used SQAE to analyze software quality.

How It Works

The tools and evaluation method behind SQAE are built upon a three-layer quality model composed of four quality areas, seven quality factors, and 76 quality attributes. Figure 4 shows how each layer in the model is constructed. Figures 5 through 8 show how each quality area is composed of quality factors.

Each factor is defined by a set of attributes. Some attributes have a numeric value, like the cyclomatic complexity, while others lead the assessor through a guided selection of a specific rating based on criteria crafted by domain experts. An example of such a question is: "Are function and variable names helpful in understanding the functionality of the code?" The choices for an assessment value are constrained to one of a few possibilities, with a maximum of five. Specific criteria for each of the possible assessment values that are articulated guide the selection process

FIGURE 4 SQAE's three-layer quality model

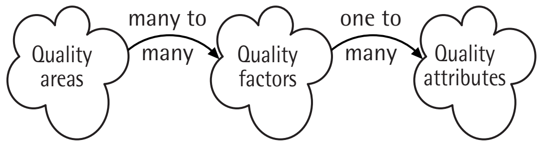
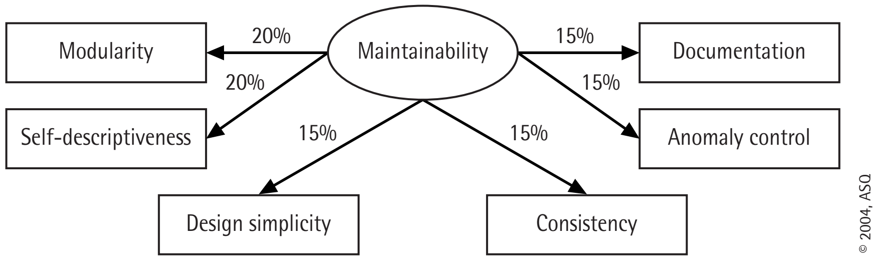


FIGURE 5 SQAE's maintainability quality area



and improve the repeatability of the process across different assessors. The guided selection of one of several predetermined possible scores for a particular attribute allows the assessment framework to include measurements of many areas of a system's architecture, design, and development guidance information that would otherwise be unassessed. Bringing these additional aspects of a system into the scope of the measurement effort provides a richer and more complete source of measurement data for identifying and measuring risk and quality issues.

As every attribute is measured, a risk profile can be built for each quality factor. An assessment of the quality at the level of the "quality areas" can be constructed from the results obtained from the quality factors.

FIGURE 6 SQAE's evolvability quality area

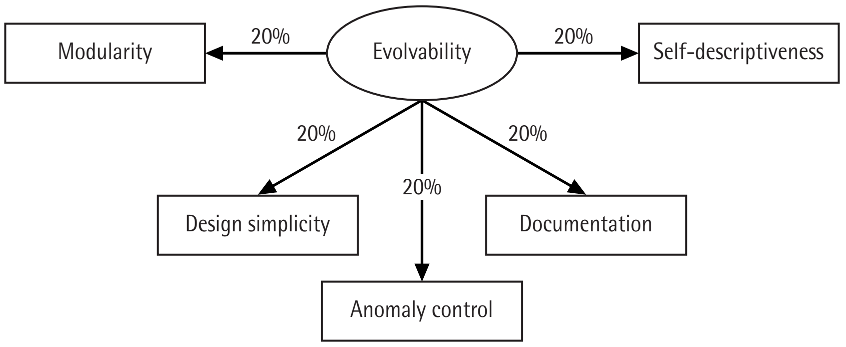


FIGURE 7 SQAE's portability quality area

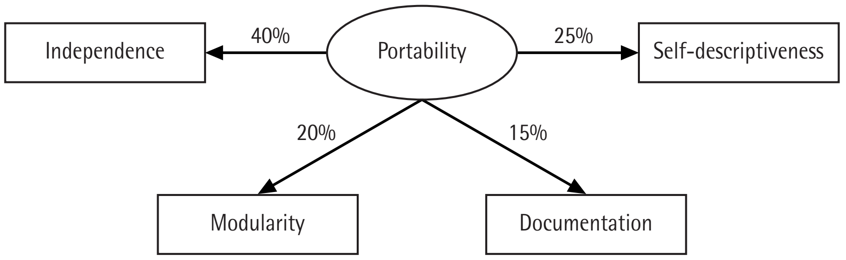
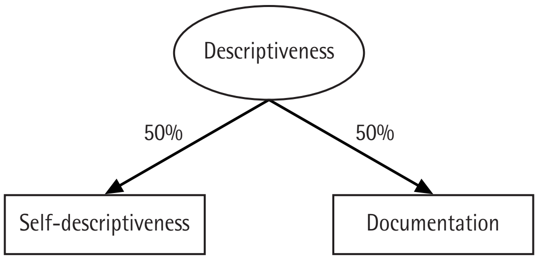


FIGURE 8 SQAE's descriptiveness quality area



Translating from SQAE to ISO/IEC 9126

The ISO/IEC 9126 standard and MITRE's SQAE have one common goal: expressing software quality, an intangible concept, in a language that is understood by all. This context is strikingly similar to a more familiar one: the context where two languages, say English and French, try to express a common concept (see Figure 9).

Since French and English both express a common concept, it is possible to translate from one to the other. The hypothesis that the same can be done with ISO/IEC 9126 and SQAE, as they also both express a common concept, is the basis of this work. As with the linguistic metaphor, it is likely that some concepts will not be easily translatable from SQAE to ISO/IEC 9126 and vice versa (see Figure 10).

FIGURE 9 French and English are used to express a common concept

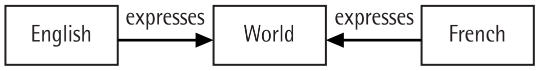


FIGURE 10 SQAE and ISO/IEC 9126 are also used to express a common concept

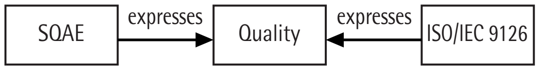
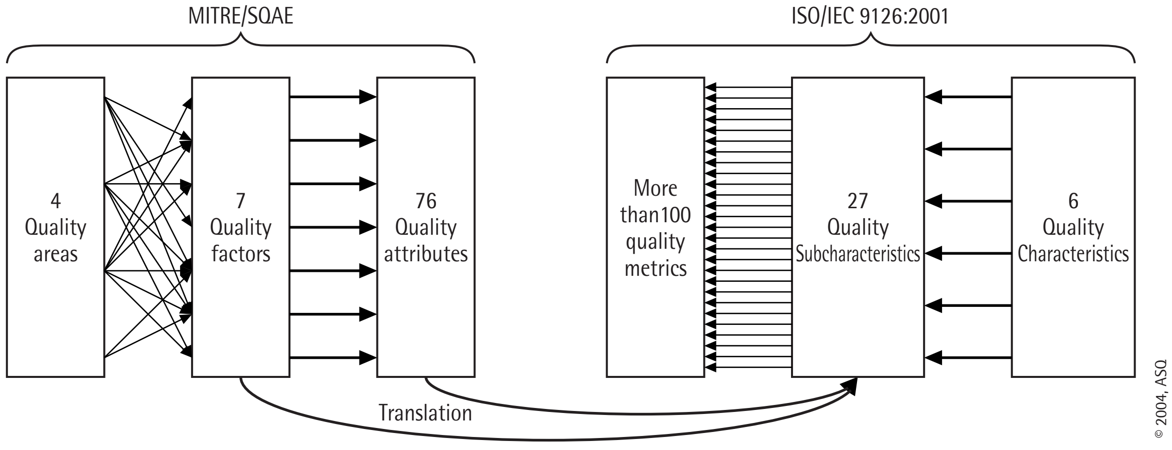


FIGURE 11 The translation activity



The translation attempt that was made has provided essential insight for the adaptation of SQAE to ISO/IEC 9126. The attempt revealed three kinds of issues:

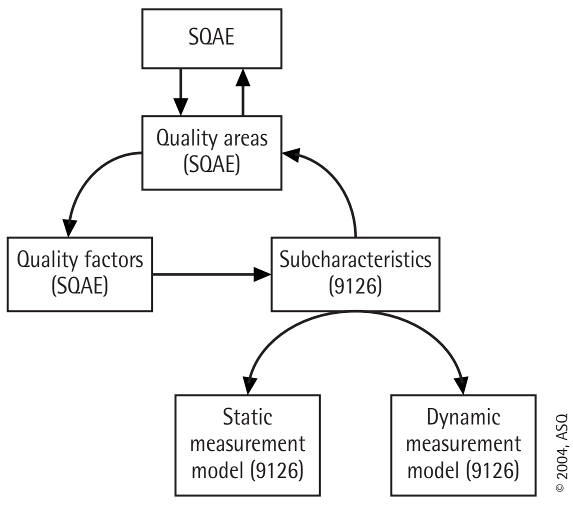
- The first possibility, and the most desirable, is when there is a perfect correspondence (translation) between a concept expressed in SQAE and ISO/IEC 9126.
- A second possibility is when a concept is not easily translatable from one language to the other. However, it might be possible to express the concept using several other concepts or by using more general ones. In such a case, a loss of precision is almost inevitable.
- A last possibility is when no translation is possible between the two languages because there is simply no common ground or because a notion is totally lacking from the target language.

The first possibility is quite probable, since both SQAE and ISO/IEC 9126 emerged from a common line of thought. The second and third possibilities are also likely, since SQAE and ISO/IEC 9126 have diverging goals. An overabundance of issues that fall into these two categories would justify a migration activity, which is defined as a set of modifications that would allow for more clarity and simplicity in expressing a given concept.

As is shown in Figure 11, the translation has been made from SQAE's quality attributes and factors to ISO/IEC 9126's subcharacteristics. It is possible to justify the choice of this level by referring to the linguistic metaphor: the results of a word-by-word translation (in this case attributes to metrics) are almost always unsatisfying. It makes more sense to take the quality attributes and factors, which respectively represent the words and the sentences of SQAE, and with them formulate the concepts embodied by ISO/IEC 9126's 27 subcharacteristics.

The results of the translation activity are based on information collected in the ISO/IEC 9126 (ISO/IEC 2001a; ISO/IEC 2003a; ISO/IEC 2003b) standard and MITRE's description of SQAE (Martin and Shaffer 1996). They are presented as a graphic showing how each quality factor and attribute contributes to the

FIGURE 12 Expressing the quality factors in terms of ISO/IEC 9126's subcharacteristics


SQAE
Quality areas (SQAE)
Quality factors (SQAE)
Subcharacteristics (9126)
Static measurement model (9126)
Dynamic measurement model (9126)
© 2004, ASQ

expression of a subcharacteristic. The following conclusions can be drawn from this work:

- All of SQAE's attributes contribute to the expression of at least one ISO/IEC 9126 subcharacteristic.
- A large number (18/27 = 66 percent) of ISO/IEC 9126's subcharacteristics are covered by SQAE's quality attributes.

Those two elements clearly paint SQAE as a language that is not as rich as ISO/IEC 9126. Although most quality characteristics are somehow evaluated by SQAE, nine are not covered, and for some others the link with SQAE's quality attributes is weak. The quality attributes are thus insufficient to completely translate SQAE to the clear and unambiguous language defined by ISO/IEC 9126.

However, the results of the translation activity clearly present the relations that exist between the two quality models and lay down the path for a migration of SQAE to ISO/IEC 9126.

FIGURE 13 Correlation between SQAE's quality factors and ISO/IEC 9126's subcharacteristics

	ISO 9126 \ SQAE	Consistency	Independence	Modularity	Documentation	Self-descriptiveness	Anomaly control	Design simplicity
Functionality	Suitability	L						
	Accuracy	H						
	Interoperability		H					
	Security							L
	Functionality compliance	L						
Reliability	Maturity						H	
	Fault tolerance						H	
	Recoverability						H	
	Reliability compliance	L					H	
Usability	Understandability	L				L		
	Learnability				H			
	Operability					H		
	Attractiveness	▨	▨	▨	▨	▨	▨	▨
	Usability compliance	L						
Efficiency	Time behavior	▨	▨	▨	▨	▨	▨	▨
	Resource utilization	▨	▨	▨	▨	▨	▨	▨
	Efficiency compliance	L						
Maintainability	Analyzability	H			H	H		H
	Changeability		H	H				H
	Stability						H	L
	Testability			H				H
	Maintainability compliance	L						
Portability	Adaptability		H					
	Installability		L					
	Coexistence				L			
	Replaceablilty				L			
	Portability compliance	L						

Legend: H: Strong correlation
L: Weak correlation

© 2004, ASQ

Migration from SQAE to ISO/IEC 9126

From the understanding gained in the translation attempt, two paths can be envisioned for the completion of the migration activity:

- Enrich the quality model behind SQAE with new quality attributes in order to make it compliant to ISO/IEC 9126.
- Express SQAE's quality factors as a composition of ISO/IEC 9126's subcharacteristics and borrow its measurement model (see Figure 12).

The first path is clearly one of brute force and not the best way to proceed. Simply adding quality attributes, factors, and areas could result in a model that is unwieldy and hard to maintain. Further modifications to ISO/IEC 9126 would inevitably result in changes to the new proposed model. The second path is akin to adding an abstraction layer between SQAE and ISO/IEC 9126 that would insulate SQAE from minor changes to ISO/IEC 9126. There are also other factors that point out the second path as the best solution:

- It will be possible to express the four quality areas, composed of seven quality factors, in the clear and unambiguous language of ISO/IEC 9126.
- By expressing the quality factors as a composition of subcharacteristics, SQAE automatically inherits from an internationally recognized set of metrics.
- It will be possible to apply the measurement methodology both statically (source code,

FIGURE 14 Consistency (SQAE) in terms of subcharacteristics (ISO/IEC 9126)

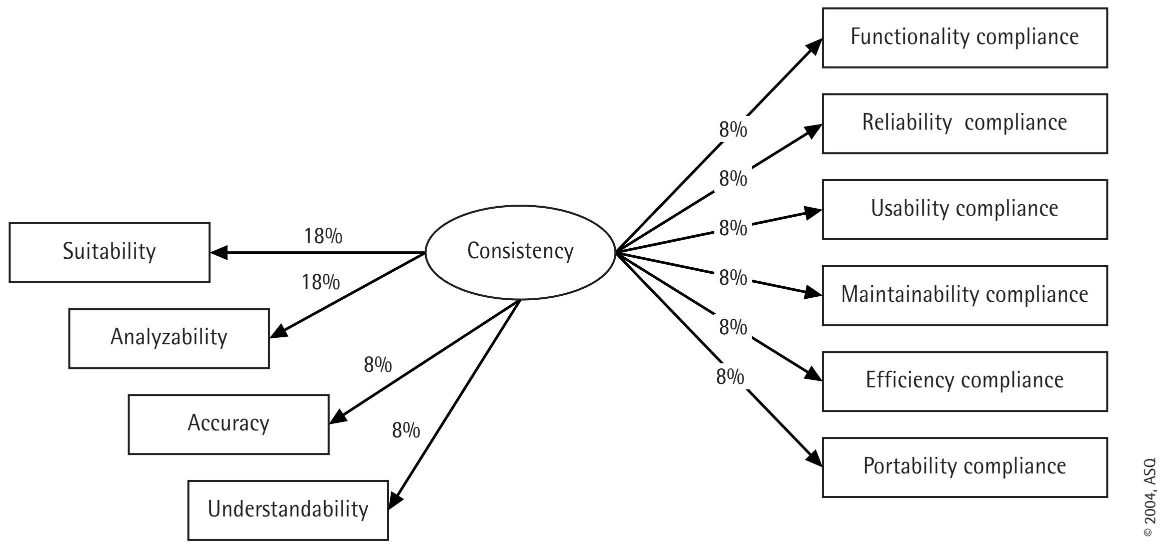
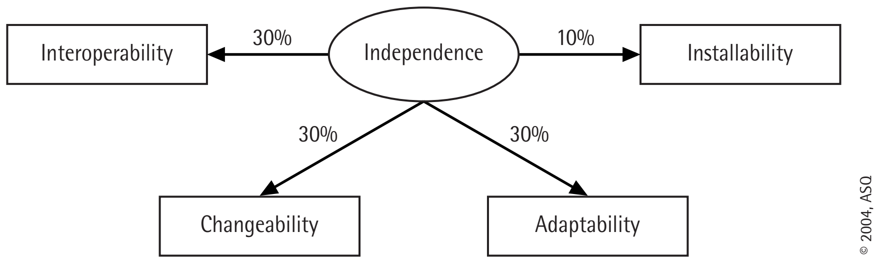


FIGURE 15 Independence (SQAE) in terms of subcharacteristics (ISO/IEC 9126)



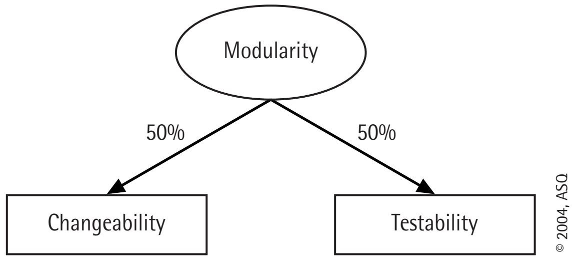
documents, and so on) and dynamically (on an executable image). This is an important advantage that will considerably enrich SQAE. This subject will be explored further in a subsequent section.

The understanding gained from the translation activity will serve as a justification of the choices that will be made during the migration.

Figure 13 presents how each quality factor can be expressed in terms of the quality subcharacteristics defined in ISO/IEC 9126. The correlation between the two models is based on the links that emerged from the translation attempt.

From Figure 13 it may be observed that each quality attribute defined in SQAE, with the exception of the ones with the crosshatch pattern, is composed of multiple subcharacteristics as defined in ISO/IEC 9126. By giving an arbitrary weight of two for a strong correlation and one for a weak correlation, the following model, illustrated in Figures 14 through 20, is obtained:

FIGURE 16 Modularity (SQAE) in terms of subcharacteristics (ISO/IEC 9126)



Since a set of well-defined metrics is associated with each subcharacteristic, quality is now measured in the clear and unambiguous way that is characteristic of ISO/IEC 9126. SQAE now becomes a method of evaluating quality rather than a way of directly measuring it. This approach is shown in Figure 21.

The proposed migration of SQAE to ISO/IEC 9126 would transform this method from one that directly measures quality to one that evaluates quality as measured by an international standard. If such a change were carried out, it would benefit SQAE in the following ways:

- The quality model and the measurements would be based on an international standard.
- The risk assessment part of SQAE would retain its value.
- If new aspects of quality are proposed, they could be integrated into the model.
- This new quality model is fully compliant with ISO/IEC 9126, because quality is now measured in a standard compliant way.

FIGURE 17 Documentation (SQAE) in terms of subcharacteristics (ISO/IEC 9126)

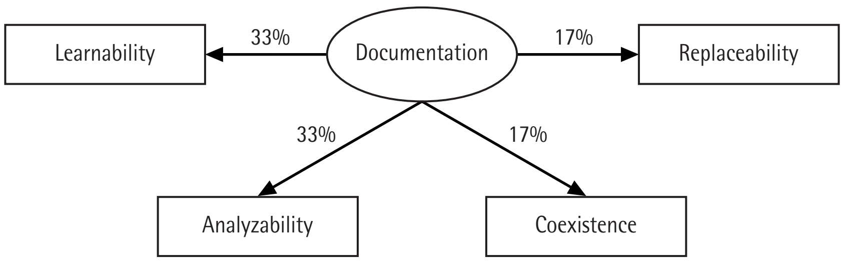


FIGURE 18 Self-descriptiveness (SQAE) in terms of subcharacteristics (ISO/IEC 9126)

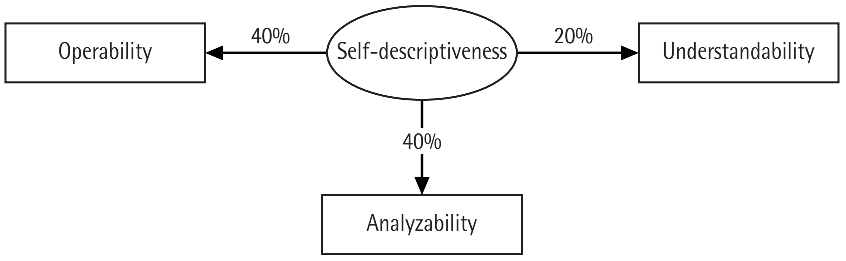
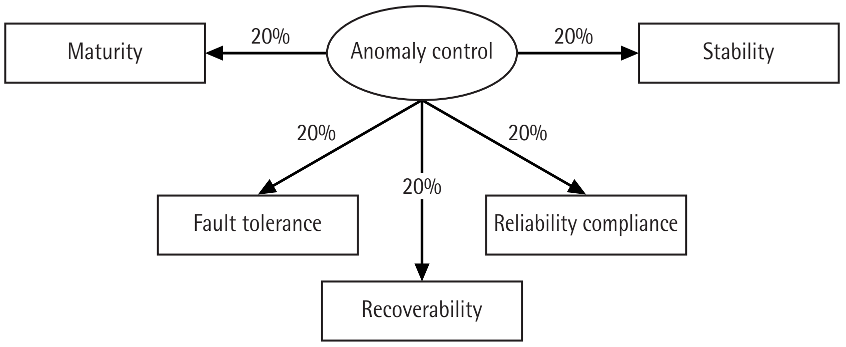


FIGURE 19 Anomaly control (SQAE) in terms of subcharacteristics (ISO/IEC 9126)



FURTHER ADAPTATIONS TO SQAE

Improvement of the Static Model

As was suggested by Figure 13, a number of subcharacteristics from ISO/IEC 9126 are not covered by SQAE. This implies that SQAE does not measure some elements of quality as defined by ISO/IEC 9126, since attractiveness, time behavior, and resource utilization are covered neither by the quality areas nor the quality factors. Such a lack of coverage is due to the fact that SQAE was designed as a method to analyze static artifacts (source code, documentation, and so on) and these subcharacteristics are naturally more prone to a static evaluation. However, ISO/IEC 9126 shows that these three subcharacteristics can indeed be measured statically. As shown in Figure 22, the following metric is associated with time behavior:

It would therefore be desirable to modify SQAE by introducing a new quality area and a few quality factors that would measure these aspects of quality. The addition of a quality area named efficiency that would measure aspects related to the efficiency of software would fill this need. Namely, we would associate with this area the following quality factors:

- Run-time efficiency: This factor would give an evaluation of the temporal efficiency of the software under evaluation.
- Interface efficiency: This quality factor would ascertain the efficiency of the software's interface.

As is the case with all the quality factors, these two new factors, shown in Figures 23 and 24, would be composed of subcharacteristics from the ISO/IEC 9126 standard:

At first glance, the new quality area, efficiency, and the associated quality factors do not seem to belong in a static quality model. Measuring the risks associated with such an attribute early on in the life cycle of a software product might lead the developers down an unfortunate path: one of early optimization. Donald Knuth (1974) once said that such a path is "the root of all evil." A method like SQAE should, however, be above such considerations and leave the decision to the evaluator. In some cases, the risks associated with efficiency must be evaluated early on. It would be interesting if SQAE could empower the evaluator as such.

FIGURE 20 Design simplicity (SQAE) in terms of subcharacteristics (ISO/IEC 9126)

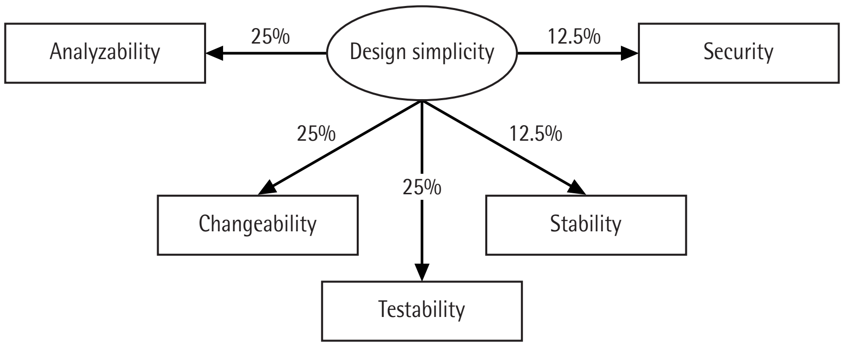
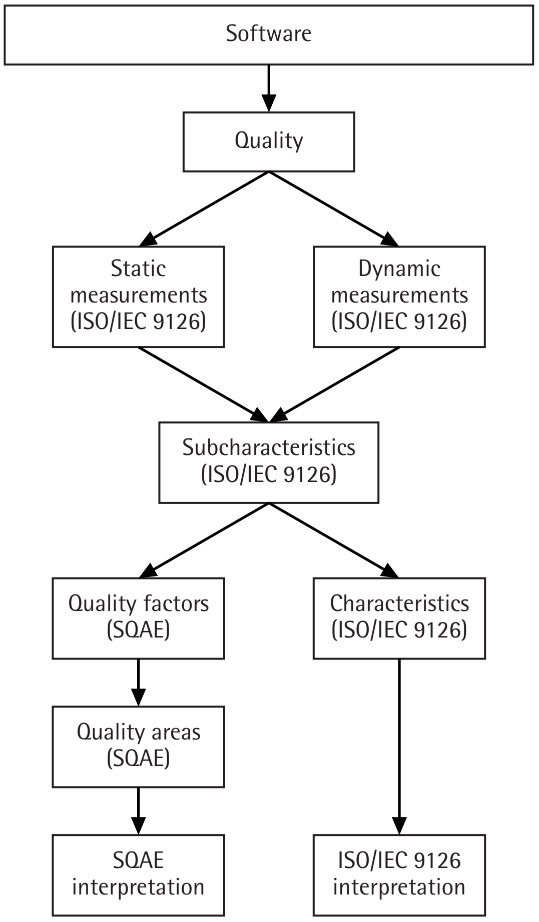


FIGURE 21 Two interpretations issued from the same measurement


Software
Quality
Static measurements (ISO/IEC 9126)
Dynamic measurements (ISO/IEC 9126)
Subcharacteristics (ISO/IEC 9126)
Quality factors (SQAE)
Characteristics (ISO/IEC 9126)
Quality areas (SQAE)
SQAE interpretation
ISO/IEC 9126 interpretation

Addition of a Dynamic Evaluation Model

SQAE was originally conceived to measure quality statically (internal quality in the terms of ISO/IEC 9126). One advantage of the method that was used to accomplish the migration is that the new model inherits the ability to measure quality dynamically (external quality in the terms of ISO/IEC 9126). This

FIGURE 22 Static metric for the time behavior subcharacteristics (ISO/IEC 2003b)

Metric name	Purpose of metric	Method of application	Formula	Interpretation of the measured value	Metric scale type	Measure
Turnaround time	What is the estimated time to complete a group of related tasks as a job lot?	Evaluate the efficiency of the operating system and the application system calls. Estimate the response time to complete a group of related tasks based on this. The following may be measured: All or parts of design specification; Test complete transaction path; Test complete modules/parts of software product; Complete software product during test phase	X=time (calculated or simulated)	The shorter the better	Ratio	X=time

is because each subcharacteristic is attached to a set of internal and external metrics. Since each quality factor is now composed of subcharacteristics, it follows that SQAE can now measure quality both statically and dynamically.

By using external metrics to measure the subcharacteristics, SQAE can now be used to give an interpretation of the external quality of the software being evaluated. It must be kept in mind that the quality model behind SQAE was created to measure internal quality. Therefore, the following question must be asked: "Do the quality areas and quality factors make sense when evaluating the dynamic aspect of quality?" A full answer to this question is a subject for the next phase of this research program. However, the examination of the documentation quality factor presented next will allow readers to observe the possible applicable trends.

In the new model that is put forward, the quality factor documentation is composed of the following subcharacteristics:

- Learnability (33 percent, strong correlation): From an external point of view, this subcharacteristic aims at measuring the learning time associated with the software's functionalities. This subcharacteristic is therefore directly related to the quality of the online documentation. A strong score in this subcharacteristic means that the software is well documented either explicitly (online help) or implicitly (intuitive interface).
- Analyzability (33 percent, strong correlation): This subcharacteristic measures the ease with which deficiencies can be detected in the software. It can be used to measure the quality of the documentation, because the ease of detecting these deficiencies will be proportional to the quality of the messages (if any) detailing them that are presented to the user.
- Coexistence (17 percent, simple correlation): The coexistence subcharacteristic measures the user's capacity to use the software being evaluated with other independent software. It

FIGURE 23 Run-time efficiency (SQAE) in terms of subcharacteristics (ISO/IEC 9126)

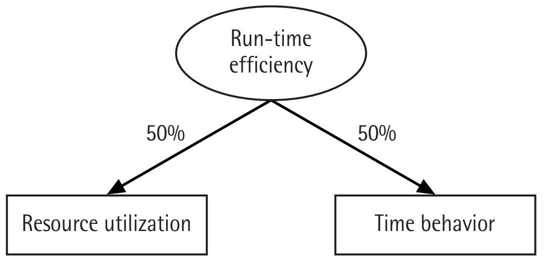
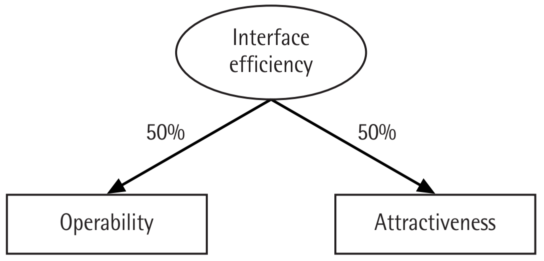


FIGURE 24 Interface efficiency (SQAE) in terms of subcharacteristics (ISO/IEC 9126)



can also measure the ease with which the software can share resources with other software running on a given system. If any coexistence problems are indicated to the user at runtime, it represents a form of documentation.

- Replaceability (17 percent, simple correlation): This subcharacteristic aims at measuring the ease with which a user can use the software under evaluation instead of another (or vice versa). For example, if a word processor helps in migrating from another word processor by providing an option to keep the key mappings, it becomes a way of documenting the functionalities in a language familiar to the user. Replaceability can thus be an indirect measure of the quality of the documentation.

The link with the documentation quality factor is undeniable for the two subcharacteristics that have a strong correlation to it. Even though this link is weaker and harder to justify for the other two subcharacteristics, it is present and measurable.

It has been shown that the documentation quality factor, which at first glance might not seem fit to a dynamic evaluation, can be used to interpret external quality. More work must be done, however, in order to validate the interpretation model for such use. Namely, the analysis that was shown previously must be carried out in more depth for each quality factor. Also, the proposed dynamic evaluation model must be validated empirically to see if the internal quality reflects the external quality as predicted by Figure 1.

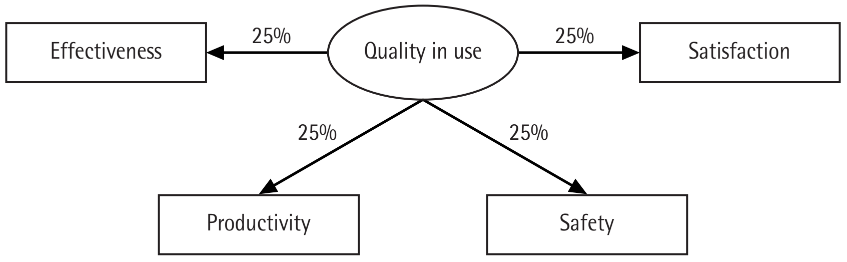
Addition of a Quality-in-Use Evaluation Model

In modern history, countless accidents could have been avoided if the interface to a system had been better thought out. In his book on the design of everyday things, Norman (2002) gives a good number of accidents (most notably the Three Mile Island incident) that could have been averted if the quality in use of some systems had been better. One of the primary failings of the first version of ISO/IEC 9126, as well as many other quality models, is the focus on the developer's view of quality at the expense of evaluating quality from the user's point of view (Pfleeger 2001). Putting too much focus on internal quality can result in systems that fail at the user interface level, as Norman points out in his book with countless examples of interface design failures.

It should be recalled, as is shown in Figure 1, that quality requirements for a system should originate in most cases from the end user. For any given software, if the user's requirements for quality in use are not met, it poses both a short-term and a long-term risk. The short-term risk emanates from a lack of acceptance by the end user of the software. It would therefore be useful to evaluate the quality in use of early prototypes in order to shape future development efforts and predict end-user acceptance. The long-term risk comes from maintenance-related problems (rework due to poor quality in use), legal liability in accidents caused by poor quality in use, and high training cost (it takes longer to train users to a nonintuitive system).

One of the most important aspects of the newest version of ISO/IEC 9126 is the integration of a quality-in-use model (ISO/IEC 2001a; ISO/IEC 2001b). As one of the main goals of SQAE is to assess the risk associated with software, it would be interesting to improve the model by including an evaluation of the risks associated with the quality in use. In order to integrate the quality-in-use model to the SQAE model, the addition of a new quality area, entitled quality in use, is needed. This new area would be composed of four new quality factors, exactly as defined by the ISO standard (ISO/IEC 2001a) and illustrated in Figure 25:

FIGURE 25 Proposed "quality-in-use" quality area



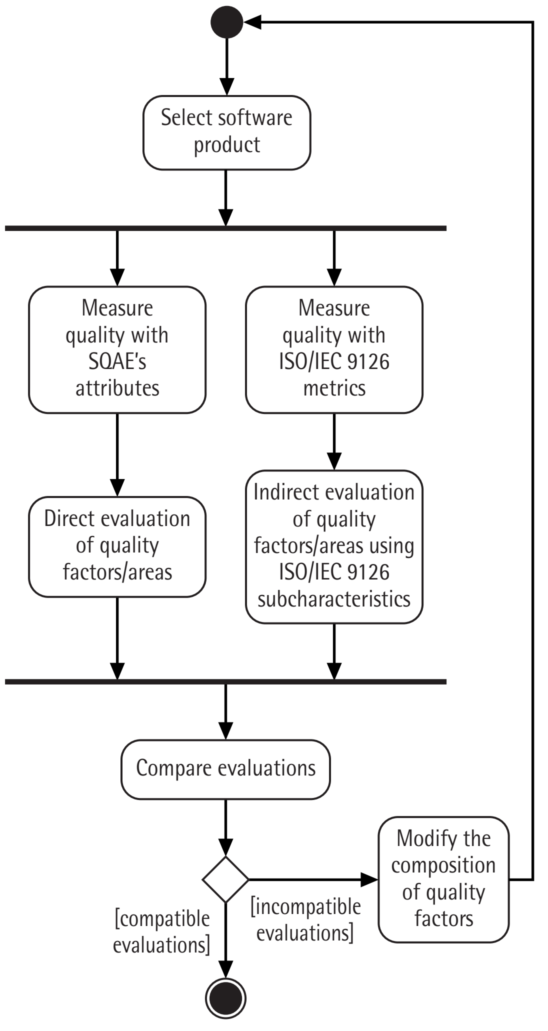
- Effectiveness: The capability of the software product to enable users to achieve specified goals with accuracy and completeness in a specified context of use.
- Productivity: The capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.
- Safety: 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.
- Satisfaction: The capability of the software product to satisfy users in a specified context of use.

CONCLUSION

By using a linguistic metaphor, it has been shown that the quality model behind MITRE's SQAE is not fully conformant to the specific details of the one proposed by the ISO/IEC 9126 standard on software product quality, although both have similar categorizations of attributes and a mapping between them is possible. The knowledge gained from the translation attempt laid down the path for a migration that would make SQAE's quality model conformant to ISO/IEC 9126. The new model now relies on every ISO/IEC 9126 subcharacteristic to define the measurable quality attributes. The proposed migration path would allow SQAE to retain its unique evaluation of the risk associated with software while gaining acknowledgment from the international community as a standard conformant method. The approach taken to perform the migration was theoretical; however, it can, and should, be validated empirically. Figure 26 illustrates a protocol that could be used to perform such a validation.

Improving quality models and tools is a never-ending task. The improvement of the evaluation model of SQAE to make it conformant to ISO/IEC 14598 is the next step to the work that was started in this article.

FIGURE 26 Validating the proposed migration path through an experimental protocol


Select software product
Measure quality with SQAE's attributes
Measure quality with ISO/IEC 9126 metrics
Direct evaluation of quality factors/areas
Indirect evaluation of quality factors/areas using ISO/IEC 9126 subcharacteristics
Compare evaluations
[compatible evaluations]
[incompatible evaluations]
Modify the composition of quality factors
© 2004, ASQ

REFERENCES

Bazzana, G., O. Andersen, and T. Jokela. 1993. ISO 9126 and ISO 9000: Friends or foes? Presented at Software Engineering Standards Symposium.

Boehm, B. W. 1978. *Characteristics of software quality*. New York: American Elsevier.

Crosby, P. B. 1979. *Quality is free: The art of making quality certain*. New York: McGraw-Hill.

Dromey, R. G. 1995. A model for software product quality. *IEEE Transactions on Software Engineering* 21: 146-162.

ISO/IEC. 2001a. ISO/IEC 9126-1: Software engineering-Software product quality—Part 1: Quality model. Geneva, Switzerland: International Organization for Standardization.

ISO/IEC. 1999. ISO/IEC 14598-1: Software product evaluation—Part 1: General overview. Geneva, Switzerland: International Organization for Standardization.

ISO/IEC. 2003a. ISO/IEC TR 9126-2: Software engineering—Software product quality—Part 2: External metrics. Geneva, Switzerland: International Organization for Standardization.

ISO/IEC. 2003b. ISO/IEC TR 9126-3: Software engineering—Software product quality—Part 3: Internal metrics. Geneva, Switzerland: International Organization for Standardization.

ISO/IEC. 2001b. ISO/IEC DTR 9126-4: Software engineering—Software product quality—Part 4: Quality in use metrics. Geneva, Switzerland: International Organization for Standardization.

Knuth, D. E. 1974. Structured programming with go to statements. *ACM Computing Surveys* 6: 261-301.

Manna, M. 1993. Maintenance burden begging for a remedy. *Datamation* (April): 53-63.

Martin, R. A. 2003. Managing software risks with metrics and measures. Presented at Second Annual Conference on the Acquisition of Software-Intensive Systems.

Martin, R. A., and L. Shaffer. 1996. *Providing a framework for effective software quality assessment*. Bedford, Mass.: MITRE Corporation.

McCall, J. A., P. K. Richards, and G. F. Walters. 1977. *Factors in software quality*. Griffiss Air Force Base, N.Y.: Rome Air Development Center Air Force Systems Command.

MITRE. 2004. The MITRE Corporation Technology Transfer Office, http://www.mitre.org/work/tech_transfer/.

Norman, D. A. 2002. The design of everyday things, first basic paperback. New York: Basic Books.

Osbourne, W. M., and E. J. Chikofsky. 1990. Fitting pieces to the maintenance puzzle. *IEEE Software* (January): 10-11.

Pfleeger, S. L. 2001. *Software engineering: Theory and practic*e, 2nd edition. Upper Saddle River, N.J.: Prentice Hall.

Pressman, R. S. 2001. *Software engineering: A practitioner's approach*, 5th ed. Boston: McGraw-Hill.

Schulmeyer, G. G., and J. I. McManus. 1998. *Handbook of software quality assuranc*e, 3rd edition. Upper Saddle River, N.J.: Prentice Hall.

Williams, R., and L. Bentrem. 2004. The diagnostic roadmap: Progress in developing an integrated view of risk identification and analysis techniques. Presented at Third Annual Conference on the Acquisition of Software-Intensive Systems.

BIOGRAPHIES

Marc-Alexis Côté is a graduate student at the École de Technologie Supérieure in Montreal, Canada. His research interests include software quality engineering, software engineering tools, and programming languages. Côté graduated *summa cum laude* in computer engineering from the University of Ottawa in 2003. He was awarded a Lucent Global Science Scholarship in 2001. He can be reached by e-mail at marc-alexis.cote@ens.etsmtl.ca .

Witold Suryn is a professor at the École de technologie supérieure, Montreal, Canada, where he teaches graduate and undergraduate software engineering courses and conducts research in the domain of software quality engineering, software engineering body of knowledge, and software engineering fundamental principles. Suryn is also the principal researcher and the director of GELOG : IQUAL, the Software Quality Engineering Research Group at École de technologie supérieure. He can be reached by e-mail at wsuryn@ele.etsmtl.ca .

Robert A. Martin has been with MITRE for 21 years and is a principal engineer in its Information Technologies Directorate. In the early 1990s Martin and his MITRE team developed a standardized software quality assessment process to help their customers improve their software acquisition and over-sight methods as well as the quality, cost, and timeliness of their delivered software products. Martin's efforts are currently focused on the interplay of cyber security, critical infrastructure protection, and software quality measurement technologies and methods. He has a bachelor's degree and a master's degree in electrical engineering from Rensselaer Polytechnic Institute and a master's of business degree from Babson College.

Claude Y. Laporte is a software engineering professor at the École de technologie supérieure. After retiring in 1992 from the Department of National Defense at the rank of major, he joined Oerlikon Contraves to coordinate the development and deployment of engineering and management processes. In 1989, he instigated the development of a software engineering center modeled on the Software Engineering Institute. As a professor at the Military College of Saint-Jean, he was also tasked to lead the development of a graduate program in software engineering for the Department of National Defense.