Usability measurement and metrics:
A consolidated model

Ahmed Seffah · Mohammad Donyaee · Rex B. Kline ·
Harkirat K. Padda

Abstract Usability is increasingly recognized as an important quality factor for interactive software systems, including traditional GUIs-style applications, Web sites, and the large variety of mobile and PDA interactive services. Unusable user interfaces are probably the single largest reasons why encompassing interactive systems – computers plus people, fail in actual use. The design of this diversity of applications so that they actually achieve their intended purposes in term of ease of use is not an easy task. Although there are many individual methods for evaluating usability; they are not well integrated into a single conceptual framework that facilitate their usage by developers who are not trained in the filed of HCI. This is true in part because there are now several different standards (e.g., ISO 9241, ISO/IEC 9126, IEEE Std.610.12) or conceptual models (e.g., Metrics for Usability Standards in Computing [MUSiC]) for usability, and not all of these standards or models describe the same operational definitions and measures. This paper first reviews existing usability standards and models while highlighted the limitations and complementarities of the various standards. It then explains how these various models can be unified into a single consolidated, hierarchical model of usability measurement. This consolidated model is called Quality in Use Integrated Measurement (QUIM). Included in the QUIM model are 10 factors each of which corresponds to a specific facet of usability that is identified in an existing standard or model. These 10 factors are decomposed into a total of 26 sub-factors or measurable criteria that are furtherdecomposed into 127 specific metrics. The paper explains also how a consolidated model, such as QUIM, can help in developing a usability measurement theory.

A. Seffah (✉) · M. Donyaee · H. K. Padda
Human-Centered Software Engineering Group, Department of Computer Science and Software
Engineering, 1455 De Maisonneuve Blvd. West, Concordia University, Montreu, Quebec, H3G 1M8,
Canada
e-mail: {seffah, donyaee padda}@cs.concordia.ca

R. B. Kline
Department of Psychology (PY 151-6), Concordia University, 7141 Sherbrooke St. West, Montreal,
Quebec, H4B 1R6, Canada
e-mail: rbkline@vax2.concordia.ca

1. Introduction

Several studies have reported the benefits of a strong commitment to usability in the software development lifecycle (e.g. Mayhew, 1999; Landauer, 1995). Among the observable benefits of usable user interfaces, one can mention human productivity and performance, safety and commercial viability. Usability is important not only to increase the speed and accuracy of the range of tasks carried out by a range of users of a system, but also to ensure the safety of the user (Repetitive Strain Injury etc.). Productivity is also imperative where the software is used to control dangerous processes. Computer magazine software reviews now include 'usability' as a ratings category. The success of commercial software may hinge on these reviews, just as the success of any software relies on the attitude of its users. Attitudes can be influenced by abstract factors such as the look and feel of a product, and how the software can be customized by the user (e.g.. colors, fonts, commands).

This explains the increasing numbers of publications in the literature have addressed the problem of how to measure software usability. Several different standards or models for quantifying and assessing usability have been proposed within the Human-Computer Interaction (HCI) and the Software Engineering (SE) communities. Examples of the latter include the ISO/IEC 9126-1 (2001) standard, which identifies usability as one of six different software quality attributes; the ISO 9241-11 (1998) standard, which defines usability in terms of efficiency, effectiveness, user satisfaction, and whether specific goals can be achieved in a specified context of use; and Directive 90/270/EEC of the Council of the European Union (1990) on minimum safety and health requirements for work with computers. However, usability has not been defined in a consistent way across the standards just mentioned or other models described momentarily. Most of these various definitions or models do not include all major aspects of usability. They are also not well integrated into current software engineering practices, and they often lack computer tool support, too.

One consequence of these weaknesses is that perhaps most software developers do not apply correctly any particular model in the evaluation of usability. This is not surprising given that there are few clear guidelines about how various definitions of usability factors, rules, and criteria are related (if at all) and how to select or measure specific aspects of usability for particular computer applications. Instead, actual practice tends to be ad hoc such that developers may employ usability methods with which they are familiar. These choices may not be optimal in many cases. That is, the effort to measure usability may be wasted without a consistent and consolidated framework for doing so. Other motivations to outline a consolidated model for usability measurement are to:

- Reduce the costs of usability testing by providing a basis for understanding and comparing various usability metrics
- Complement more subjective, expert-based evaluation of usability
- Provide a basis for clearer communication about usability measurement between software developers and usability experts
- Promote sound usability measurement practices that are more accessible to software developers who may not have strong backgrounds in usability engineering

Table 1 Usability attributes of various standards or models

Constantine & Lockwood (1999)	ISO 9241-11 (1998)	Schneiderman (1992)	Nielsen (1993)	Preece et al. (1994)	Shackel (1991)
Efficiency in use	Efficiency	Speed of performance	Efficiency of use	Throughput	Effectiveness (Speed)
Learnability		Time to learn	Learnability (Ease of learning)	Learnability (Ease of learning)	Learnability (Time to learn)
Rememberability		Retention over time	Memorability		Learnability (Retention)
Reliability in use		Rate of errors by users	Errors/safety	Throughput	Effectiveness (Errors)
User satisfaction	Satisfaction (Comfort and acceptability of use)	Subjective satisfaction	Satisfaction	Attitude	Attitude

2. Definition and perceptions of usability as a quality factor

As mentioned, the term *usability* has been defined in different ways in the literature, which makes it a confusing concept. To illustrate this point, some broad definitions of *usability* from three different standards are listed next:

- "A set of attributes that bear on the effort needed for use and on the individual assessment of such use, by a stated or implied set of users" (ISO/IEC 9126, 1991)
- "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" (ISO 9241−11, 1998)
- "The ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component" (IEEE Std.610.12-1990)

There are also varying definitions across different sets of standards or authors concerning more specific attributes (facets, aspects, factors) of usability. Some of these definitional sets are summarized in Table 1. Each row in the table lists areas of apparent agreement concerning attributes of usability. For example, all sources in Table 1 describe "efficiency" as a usability attribute, although not all sources use this particular term (e.g., Nielsen, 1993; Schneiderman, 1992). Also, not all sources in Table 1 include the same core set of usability attributes. Both characteristics just mentioned can be a source of confusion for researchers and developers alike. These examples highlight the need for a consolidated model about usability measurement with consistent terms for usability attributes and metrics.

3. Review of existing usability measurement standards or models

The general domain of measurement in science dates back many years (e.g., Stevens, 1959). In contrast, the systematic consideration of measurement in the area of software quality is much more recent (e.g., Fenton and Whitty, 1995) but, of course, this is in part due to the relative youth of this field. In particular, usability measurement today is not generally

concerned with similar research in software engineering measurement: the relating of data about measurable attributes to hypothetical constructs that in software engineering concern quality attributes such as reusability or reliability (Curtis, 1980). In this section, we review existing usability measurement standards or models while highlighting some of their contributions and limitations.

3.1. Usability in ISO standards related to HCI

There are three major ISO/IEC standards for quantifying and measuring usability. The ISO 9241-11 (1998) standard identified efficiency, effectiveness, and satisfaction as major attributes of usability. The major limitation of this standard as a quality model is that it is too abstract. For example, a reader might assume that because a metric is listed for a factor, then by default the standard states that the two are related. However, this standard actually gives little discussion about this issue. The same standard also gives relatively few guidelines about how to interpret scores from specific usability metrics.

More recently, usability has been defined in the ISO/IEC 9126-1 (2001) standard as a software quality attribute that can be decomposed into five different factors, including understandability, learnability, operability, attractiveness, and usability compliance with published style guides or conventions for user interfaces. The ISO/IEC 9126-4 (2001) standard defined the related concept of *quality in use* as a kind of higher-order software quality attribute that can be decomposed into three factors, effectiveness (i.e., usefulness), productivity, and safety. Some software cost estimation methods, such as COCOMO II (Boehm et al., 2000), take account of the impact of the software development environment on the areas just mentioned.

In the ISO/IEC 9126-4 (2001) standard, the difference between usability and the quality in use is a matter of context of use. Specifically, when usability is evaluated, the focus is on improving the user interface while the context of use is treated as a given. This means that the level of usability achieved will depend on the specific circumstances in which the product is used. In contrast, when quality in use is evaluated, any component of context of use may be subject to change or modification. This is why Bevan and Macleod (1994) viewed usability as a result of use of a computer tool in a particular context. Specifically, they assume that quality in use can be measured as the outcome of interactions with a computer system, including whether intended goals of the system are achieved (effectiveness) with the appropriate expenditure of resources (e.g., time, mental effort) in a way the user finds acceptable (satisfaction).

The ISO/IEC 9126-4 (2001) standard also distinguished external versus internal quality and defined related metrics. Internal quality generally concerns properties of the non-executable portion of a software product during its development, and metrics for internal quality generally concern the quality of intermediate deliverables, such as the source code for a prototype version. In contrast, external quality concerns the behavior of the computer system of which the software product is a part. Metrics for external quality can be obtained only by executing the software product in the system environment for which the product is intended. Metrics in ISO/IEC 9126-4 (2001) are also classified in terms of level of measurement (i.e., nominal, ordinal, interval, or ratio) and measurement type (e.g., count variables [the number of times an event occurs], elapsed time, size [source size, function size, etc.]).

The ISO/IEC 14598-1 (1999) standard suggested a model for measuring quality in use from the perspective of internal software quality attributes. Along these lines, Bevan and Azuma (1997) conceptualized internal quality attributes as the cause and quality in use factors as the effects. This standard also assumes that the user's needs in terms of usability goals are expressed as a set of requirements for the behavior of the product in use. This concerns

external quality attributes of the software product when it is executed. These requirements should be expressed as metrics that can be measured when the software is used in a computer system in its intended context. Various metrics for external quality in this standard concern effectiveness, efficiency, and user satisfaction.

3.2. Usability in traditional software quality models

The idea of usability has been represented in various software engineering quality models for at least three decades. For example, McCall, Richards, and Walters (1977) described one of the earliest of these models referred to as the GE (General Electric) model or FCM (Factors, Criteria, Metrics) model. This hierarchical quality model was made up of a total of 11 quality factors, 25 quality criteria, and 41 specific quality metrics. In this model, quality factors are hypothetical constructs that correspond to the external view of the system as perceived by its users. A hypothetical construct as defined by Nunnally and Bernstein (1994) reflects the hypothesis that a variety of specific measures will correlate with one another or will be similarly affected by experimental manipulation. Hypothetical constructs, such as "software usability" and "software comprehension," are not directly measurable. Instead, they can be only inferred indirectly through observed measures, such as those for perceived effectiveness, user satisfaction and performance evaluation.

Usability in McCall's model is decomposed into three criteria, operability, training, and effectiveness. A quality criterion can be related to more than one factor and is directly measurable with specific metrics. Similar to McCall, Boehm et al.'s (1978) quality model is also hierarchical. It assumes that quality attributes are higher-order characteristics (hypothetical constructs) that are not directly measurable. Also similar to McCall's model, quality attributes (factors) are decomposed into quality criteria, which in turn are decomposed into directly measurable characteristics (metrics). Boehm's model incorporates 19 different quality factors encompassing product utility, maintainability, and portability. However, specific metrics in Boehm's model are not always associated with the same quality factors as in McCall's models, which can cause confusion.

3.3. Other specific measurement models

Besides the standards and models described in the previous sections, some other models and tools for evaluating usability have been proposed over last 15 years. Some of the most influential works in this area are described next.

The Metrics for Usability Standards in Computing (MUSiC; Bevan, 1995; Macleod et al., 1997) model was concerned specifically with defining measures of software usability, many of which were integrated into the original ISO 9241 standard. Examples of specific usability metrics in the MUSiC framework include user performance measures, such as task effectiveness, temporal efficiency, and length or proportion of productive period. However, a strictly performance-based view of usability cannot reflect other aspects of usability, such as user satisfaction or learnability. As part of the MUSiC project, a 50-item user satisfaction questionnaire called the Software Usability Measurement Inventory (SUMI; Kirakowski and Corbett, 1993) was developed to provide measures of global satisfaction and of five more specific usability areas, including effectiveness, efficiency, helpfulness, control, and learnability.

The Skill Acquisition Network (SANe; Macleod, 1994) model dealt with the analysis of the quality of use of interactive devices. This approach assumes a user interaction model that defines user tasks, the dynamics of the device, and procedures for executing user tasks. Specifically, a task model and a device model are simultaneously developed and subsequently

linked. Next, user procedures are simulated within the linked task-device model. A total of 60 different metrics are described in this framework, of which 24 concern quality measures. Scores from the latter are then combined to form a total of five composite quality measures, including:

- Efficiency, which is determined by the estimated costs (e.g., total time) of executing user procedures
- Learning, which is defined as the number of states and state transitions necessary to carry out user tasks
- Adaptiveness, which concerns the functionality of the device within a given application domain
- Cognitive workload, which is determined by the controllability of the application, decision complexity (alternatives from which the user can choose), and memory load
- Effort for error correction, which concerns the robustness of a device and the costs for error recovery

The semi-Automated Interface Designer and Evaluator (AIDE, Sears, 1995) model provided a software tool to evaluate static HTML pages according to a set of predetermined guidelines about Web page design. These guidelines concern things such as the placement and alignment of screen elements (e.g., text, buttons, or links). The AIDE tool can also generate alternative interface layouts and evaluate some aspects of a design. Designs are evaluated in AIDE according to both task-sensitive metrics and task-independent metrics. Task-sensitive metrics incorporate task information into the development process, which may ensure that user tasks guide the semantics of interface design. Task-independent metrics tend to be based on principles of graphic design and help to ensure that the interface is aesthetically pleasing. Altogether the AIDE tool can measure a total of five different usability metrics, including efficiency, alignment, horizontal balance, vertical balance, and designer-specified constraints (e.g., element positioning).

The Diagnostic Recorder for Usability Measurement (DRUM; Macleod and Rengger, 1993) is a software tool for analyzing user-based evaluations and the delivery of these data to the appropriate party, such as a usability engineer. The Log Processor component of DRUM is the tool concerned with metrics. It calculates several different performance-based usability metrics, including:

- Task time, or the total time required for each task under study
- Snag, help, and search times, which concern the amount of time users spend dealing with problems such as seeking help or unproductively hunting through a system
- Effectiveness, which as a metric is derived from measures of the quantity and quality of task output and measures whether users succeed in achieving their goals when working with a system
- Efficiency, which relates effectiveness to the task time and thus measures the rate of task output
- Relative efficiency, which indicates how efficiently a task is performed by a general user compared with an expert user on the same system or with the same task on another system
- Productive period, or the proportion of task time *not* spent in snag, help, or search (i.e., the relative amount of productive work time)

The Goals, Operators, Methods, and Selection rules (GOMS; John and Kieras, 1996) model for a particular task consists of descriptions of the methods needed to accomplish specified goals with a software system. The methods are a series of steps consisting of operations that the user must perform with that system. A method may call for sub-goals to

be accomplished, so methods have a hierarchical structure in the GOMS framework. If more than one method is needed in order to accomplish a goal, then the model outlines selection rules that can be used to choose the appropriate method depending on the context. By adding estimated time to accomplish a task as part of the description of a task, the GOMS model can be used to predict aspects of the expert user performance, such as total task time. This predictive model can be useful during the task analysis phase, but a fully-functional system is eventually needed to collect measurements that verify the predictions.

The National Institute of Standards and Technology (NIST) Web Metrics (Scholtz and Laskowski, 1998) is a set of six computer tools and several metrics that support rapid, remote, and automated testing and evaluation of website usability. In their study on empirically-validated web page design metrics, Ivory and Hearst (2001) found that usability prediction with the help of metrics matches in some cases up to 80% of the results based on expert evaluation of the same Web pages.

4. Rationale for a consolidated model

There are at least three major reasons that for a consolidated model of usability measurement.

The hierarchical quality models just described have some common limitations. They are all rather vague in their definitions of the lower-level usability metrics needed in order to obtain satisfactory measures of higher-order software quality factors. For example, there is relatively little information about how to apply or interpret scores from specific quality metrics. There is relatively little guidance about ways to account for the degree of influence of individual quality factors, such as the role of learnability versus understandability in usability problems. This lack of consistent operational definitions can make it difficult to compare results across different usability studies. There is also comparatively little information about exactly how to select a set of usability factors or metrics considering things such as management objectives, business goals, competition, economics, or resource limitations on product development. These problems could potentially be addressed through a better tool support for use by individuals who are given the responsibility to evaluate usability. A consolidated model can also address this problem by incorporating different viewpoints on usability and its measurement in a consistent, uniform way.

Second, most of the software quality models just described, including the ISO/IEC 9126 standard, are static. That is, none of these models really describes the relation between phases in the product development cycle and appropriate usability measures at specific project milestones. It is important to be able to relate software measures to project tracking and to target values at the time of delivery of the software. Also, none of these models typically give any guidance concerning the application of usability measures and attributes in the identification and classification of risk (Hyatt and Rosenberg, 1996).

Third, it can be rather difficult to apply usability standards in practice, that is, to decide exactly how to measure the usability of a particular application. Specifically, it is not always clear how usability factors, criteria, and metrics defined in various standards or models are related or whether one set of metrics may be more advantageous than others. A consolidated model should support the exploration of the relations among sets of factors, criteria, and metrics, again in a consistent and clear way. This characteristic should also help to ameliorate the problem that the successful application of usability standards in practice often seems to depend on idiosyncratic factors, such as the skill of individual practitioners. As in other engineering disciplines, it is better that the application of a particular method—in this case usability measurement—be algorithmic such that most practitioners can successfully use

the method. Also, a consolidated model should be flexible and generic enough in order to help developers or testers who may not be already familiar with usability metrics to create a concrete, step-by-step measurement plan for different types of applications. Examples where usability issue may be critical include safety-critical systems or Web applications where accessibility is crucial.

Finally and most importantly, a consolidated measurement framework should also provide guidelines for interpretation of the data collected under it. This characteristic is essential in order to allow persons who are not already usability engineers, such as developers or quality assurance managers, to make informed decisions about how to evaluate the usability of particular applications in a specific context. This built-in support for interpretation may help to avoid the problem that occurs when data from multiple usability metrics are collected, but then developers are not really sure about what the numbers actually mean.

5. QUIM: A roadmap for a consolidated model

We propose QUIM (Quality in Use Integrated Measurement) as a consolidated model for usability measurement. Similar to the existing software engineering models and most usability measurement frameworks described earlier, QUIM is hierarchical in that it decomposes usability into factors, then into criteria, and finally into specific metrics. In this sense, QUIM follows the IEEE 1061 (1998) standard (Software Quality Metrics Methodology), which outlines methods for establishing quality requirements as well as identifying, implementing, analyzing, and validating both process and product quality metrics (see Schneidewind, 1992; Yamada et al., 1995). The main application for QUIM at this time is to provide a consistent framework and repository for usability factors, criteria, and metrics for educational and research purposes. After empirical validation of the hierarchical relationships implied by QUIM, it may be possible to create an application-independent ontology about usability measurement (see Jarrar et al., 2003). By instantiating such an ontology, it may be possible to create a knowledge base that can be used for usability prediction, that is, as an automated quality assessment tool that reduces design, testing, and maintenance time.

The QUIM framework serves basically as a consolidated model under which other models for usability measurement can be derived. It essentially combines the existing models described in Section 3. The QUIM model decomposes usability into factors, criteria and metrics. In contrast to other hierarchical models described earlier, QUIM has two explicit supplementary levels, data and data collection methods. Data are elements of usability metrics, that is, they are quantities that are combined in the function that define the metric. By themselves, data are not generally interpretable as a measure of some facet of usability. For example, the number of interface objects visible on the screen could be considered a datum. A usability metric based in part on this datum could be the proportion of these objects that are actually relevant to a particular task.

5.1. User model, context of use and usability in context

As defined in section 2, usability is generally a relative measure of whether a software product enables a particular set of users to achieve specified goals in a specified context of use. This means that usability can vary from a user to another one and in general with the context of use. This includes user profiles (i.e., who are the users), task characteristics, hardware (including network equipment), software, and physical or organizational environments. Therefore, the measurement of usability requires that we should know in advance the characteristics of the

Table 2 Examples of context of use attributes from ISO 9241-11 (1998)

Component	Relevant data
Users characteristics	Psychological attributes including cognitive style (e.g., analytic vs. intuitive, attitude towards the job, motivation to use the system, habits and motor-sensory capabilities)
	Knowledge and experience including native language, typing skills, education and reading level, system experience (e.g. knowledge of computer and OS), task experience (e.g., knowledge of the domain), application experience (e.g., knowledge of similar applications)
Task characteristics	Frequency
	Duration and importance
	Task flexibility/pacing
	Physical and mental demands
	Complexity as perceived by the user Task structure (e.g., highly structured vs. unstructured) Task flow including input/start conditions, output/finish conditions, and dependencies
	Relation to business workflow
Technical environment characteristics	Hardware capabilities and constraints
	Network connection
	Operating system
	Supporting software
Physical environment	Noise level, privacy, ambient qualities, potential health hazards, and safety issues
Organizational environment	Structure of the operational teams and the individual staff members' level of autonomy
	Work and safety policies
	Organizational culture
	Feedback to employees on job quality
Social environment	Multi- or single-user environment
	Degree of assistance available
	Interruptions (e.g., disturbing environment)

target users and the kinds of tasks they will carry out with the system. Lack of knowledge about either users or tasks may lead to the inability to formulate a realistic usability measurement plan. It may also result in a product that only by chance happens to meet user needs. Information about the context of use (users, tasks, settings, etc.) should be documented as part of the requirement specifications for the application. Summarized in Table 2 are context of use attributes from the ISO 9241-11 (1998) standard. These attributes concern characteristics of users and tasks and of technical, physical, social, or. It should be noted that not all of the attributes listed in Table 2 will be relevant for all types of usability tests. It may also be necessary to consider additional attributes for a given type of interactive system such as for mobile applications or a specific type of test such user performance measurement.

Furthermore, context of use descriptions should be established iteratively at the early stages to develop measurement benchmarks and requirements. Later, when some kind of workable prototype is available for evaluation, the context of use is considered when selecting the aspects of the system that should be measured. In this way, the consideration of

context in usability measurement will ideally make such measurement more realistic and meaningful.

5.2. Usability factors

The earlier analysis (Section 3) indicated that the ISO 9241-11 (1998) standard takes a broader perspective on usability measurement than the ISO/IEC 9216 (2001) standards. The former standard focuses on tasks and environment questions as organizational factors, and its usability definitions concern software quality characteristics, which are distinct from those of usability in the ISO 9126 standards, such as functionality, precision and effectiveness. All these characteristics contribute to software quality. However, these viewpoints in the two different standards are complementary. According to Bevan and Schoeffel (2001), an interactive computer system does not have intrinsic usability, only an ability to be used in a particular context of use. From this perspective, the ISO 9241-11 standard can help us to understand in which context particular attributes specified in the ISO/IEC 9126 standards are required. The ISO 9241-11 standard is also generally recognized in Europe (e.g., it is published in Germany as a Deutsches Institut für Normung [DIN] standard). For these reasons, we selected the basic ISO 9241-11 standard as the baseline for QUIM.

Usability is closely related to ease of learning or learnability without necessarily implying a high performance in task execution. For example, experienced users may be interested mainly in completing a wider range and number of tasks with minimal obstruction than with quickly completing a smaller range of tasks. This is in part why some researchers have proposed their own usability measurement models. This is sometimes done by adding usability definitions or attributes, such as learnability, to those of the basic ISO 9241-11 standard. Also included in QUIM are some emerging usability factors, those identified only quite recently as being important considerations in some contexts. For example, the inclusion of trustfulness as a usability factor in QUIM was based in part on the Cheskin Research and Studio Archetype/Sapient (1999) report on user confidence in e-commerce sites. The recent emphasis on developing Web sites that are accessible to special kinds of users, such as disabled persons, motivated the inclusion of the universality and accessibility factors in QUIM (e.g., Olsina et al., 1999). Other researchers and some other standards, such as ISO 13407 (1999) and IEC 300-3-9 (2004), include human security as a usability characteristic for certain critical systems, such as medical equipment or nuclear power stations. Security as a basic usability characteristic is also included in QUIM.

Based on our review of the usability measurement literature, the 10 usability factors briefly described next are included in the QUIM consolidated model:

1. **Efficiency**, or 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.
2. **Effectiveness**, or the capability of the software product to enable users to achieve specified tasks with accuracy and completeness.
3. **Productivity**, which is the level of effectiveness achieved in relation to the resources (i.e. time to complete tasks, user efforts, materials or financial cost of usage) consumed by the users and the system. In contrast with efficiency, productivity concerns the amount of useful output that is obtained from user interaction with the software product. Macleod et al. (1997) noted that there are generally two types of user task

actions, one that is productive and the other is unproductive. The productive user task actions are those that contribute to the task output. Therefore, our definition of productivity considers the productive resources that are expended in accomplishing users' tasks.

4. **Satisfaction**, which refers to the subjective responses from users about their feelings when using the software (i.e., is the user satisfied or happy with the system?). Reponses from users are generally collected using questionnaire such as the SUMI we listed in section 3.4.

5. **Learnability**, or the ease with which the features required for achieving particular goals can be mastered. It is the capability of the software product to enable users to feel that they can productively use the software product right away and then quickly learn other new (for them) functionalities.

6. **Safety**, which concerns whether a software product limits the risk of harm to people or other resources, such as hardware or stored information. It is stated in the ISO/IEC 9126-4 (2001) standard that there are two aspects of software product safety, operational safety and contingency safety. Operational safety refers to the capability of the software product to meet the user requirements during normal operation without harm to other resources and the environment. Criteria to be considered in the evaluation of operational safety include consistency, accuracy, completeness, security, and insurance. Contingency safety concerns the capability of the software product to operate outside its normal operation but still prevent risks. Criteria for contingency safety include fault tolerance and resource safety.

7. **Trustfulness**, or the of faithfulness a software product offers to its users. This concept is perhaps most pertinent concerning e-commerce websites (e.g., Ahuja, 2000; Atif, 2002; Cheskin Research & Studio Archetype/Sapient, 1999; Friedman et al., 2000; Tilson et al., 1997), but it could potentially apply to many different kinds of software products.

8. **Accessibility**, or the capability of a software product to be used by persons with some type of disability (e.g., visual, hearing, psychomotor). The World Wide Web Consortium (Caldwell et al., 2004) suggested various design guidelines for making Web sites more accessible to persons with disabilities.

9. **Universality**, which concerns whether a software product accommodates a diversity of users with different cultural backgrounds (e.g., local culture is considered).

10. **Usefulness**, or whether a software product enables users to solve real problems in an acceptable way. Usefulness implies that a software product has practical utility, which in part reflects how closely the product supports the user's own task model. Usefulness obviously depends on the features and functionality offered by the software product. It also reflects the knowledge and skill level of the users while performing some task (i.e., not just the software product is considered).

The ten factors just described are not assumed to be independent, which is not the case in some existing hierarchical models of usability measurement. In e-commerce, for example, online customers may trust a site only if they feel both safe and satisfied when using it. Also, some terms related to usability, such as usefulness and learnability, are represented in QUIM as factors and not as criteria as in some existing hierarchical models. Other potential usability factors may be added to future versions of QUIM, such as:

- Portability or the capacity of a system to be displayed in different platforms,
- Adaptability or the capacity of a system to be adapted or to adapt itself to the context and
- Comprehension

5.3. Measurable criteria

Each factor in QUIM is broken down into measurable criteria (sub-factors). A criterion is directly measurable via at least one specific metric. Presented in Table 3 are definitions of the 26 criteria in QUIM. These definitions all assume a particular context of use or stated conditions for a software feature.

Summarized in Table 4 are the relations between the 10 usability factors in QUIM (Section 5.2) and the 26 usability criteria (Table 3). These relationships were derived based on rational analysis of the factors and the criteria. For example, the efficiency factor is assumed to correspond to criteria such as resource utilization, operability, and feedback, among others listed in Table 4. That is, we hypothesize that efficiency can be measured with metrics associated with the criteria listed for this factor in the table. Looking at Table 4 from the perspective of the criteria, we hypothesize that minimal memory load, for example, is related to at six different factors, including efficiency, satisfaction, and learnability, among others. As mentioned, the relations represented in Table 4 are hypotheses, and these hypotheses require further evaluation. In this sense, the QUIM model provides a starting for the eventual validation of these hypotheses.

5.4. Metrics

Based on our review of existing usability measurement standards and models, we identified a total of 127 specific usability metrics. Some metrics are basically functions that are defined in terms of a formula, but others are just simple countable data. Countable metrics may be extracted from raw data collected from various sources such as log files, video observations, interviews, or surveys. Examples of countable metrics include the percentage of a task completed, the ratio of task successes to failures, the frequency of program help usage, the time spent dealing with program errors, and the number of on-screen user interface elements.

Calculable (refined) metrics are the results of mathematical calculations, algorithms, or heuristics based on raw observational data or countable metrics. For example, a proposed formula by Bevan and Macleod (1994) for calculating task effectiveness is:

$$TE = \text{Quantity} \times \text{Quality}/100$$

Where Quantity is the proportion of the task completed and Quality is the proportion of the goal achieved. The proportions just mentioned are the countable metrics that make up the calculable TE metric. Listed in Table 5 are examples of additional calculable metrics included in QUIM.

Depending on the phase of the software life cycle in which they are applied, usability metrics can be classified into one of two major categories, testing and predictive. Data from testing metrics are collected in order to measure the actual use of working software while identifying the problems encountered. Collection of these data requires that a fully-functional system or high-fidelity prototype is available. A relatively large set of testing metrics have been proposed in the usability literature. These include preference metrics, which quantify subjective evaluations, preferences, and level of satisfaction of end users; and performance metrics, which measure the actual performance of the users when accomplishing a task in a certain context (e.g., success rate, error rate, time completion time). A predictive metric (also called a design metric) is a value that describes some aspect of that may provide an estimate or prediction of system usability. There is usually an explicit claim associated with a predictive metric. Suppose that a larger font size (up to a certain point) is expected to be

Table 3 Usability criteria in the QUIM model.

Criteria	Description
Time behavior	Capability to consume appropriate task time when performing its function
Resource utilization	Capability to consume appropriate amounts and types of resources when the software performs its function (ISO/IEC 9126-1, 2001)
Attractiveness	Capability of the software product to be attractive to the user (e.g., through use of color or graphic design; ISO/IEC 9126-1, 2001)
Likeability	User's perceptions, feelings, and opinions of the product (Rubin, 1994)
Flexibility	Whether the user interface of the software product can be tailored to suit users' personal preferences
Minimal action	Capability of the software product to help users achieve their tasks in a minimum number of steps
Minimal memory load	Whether a user is required to keep minimal amount of information in mind in order to achieve a specified task (Lin et al., 1997)
Operability	Amount of effort necessary to operate and control a software product
User guidance	Whether the user interface provides context-sensitive help and meaningful feedback when errors occur
Consistency	Degree of uniformity among elements of user interface and whether they offer meaningful metaphors to users
self-descriptiveness	Capability of the software product to convey its purpose and give clear user assistance in its operation
Feedback	Responsiveness of the software product to user inputs or events in a meaningful way
Accuracy	Capability to provide correct results or effects (ISO/IEC 9126−1, 2001)
Completeness	Whether a user can complete a specified task
Fault tolerance	Capability of the software product to maintain a specified level of performance in cases of software faults or of infringement of its specified interface (ISO/IEC 9126–1, 2001)
Resource safety	Whether resources (including people) are handled properly without any hazard
Readability	Ease with which visual content (e.g., text dialogs) can be understood
Controllability	Whether users feel that they are in control of the software product
Navigability	Whether users can move around in the application in an efficient way
Simplicity	Whether extraneous elements are eliminated from the user interface without significant information loss
Privacy	Whether users' personal information is appropriately protected
Security	Capability of the software product to protect information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access (ISO/IEC 12207, 1995)
Insurance	Liability of the software product vendors in case of fraudulent use of users' personal information
Familiarity	Whether the user interface offers recognizable elements and interactions that can be understood by the user
Load time	Time required for a Web page to load (i.e., how fast it responds to the user)
Appropriateness	Whether visual metaphors in the user interface are meaningful

Table 4 Relations between factors and criteria in QUIM.

Criteria	Efficiency	Effectiveness	Satisfaction	Productivity	Learnability	Safety	Trustfulness	Accessibility	Universality	Usefulness
Time behavior	+				+					
Resource utilization	+				+					+
Attractiveness			+						+	
Likeability			+							
Flexibility		+	+					+	+	+
Minimal action	+		+		+			+		
Minimal memory load	+		+		+			+	+	+
Operability	+		+				+	+		+
User guidance			+		+			+	+	
Consistency		+			+	+		+	+	
Self-descriptiveness					+		+	+	+	
Feedback	+	+							+	+
Accuracy		+				+				+
Completeness		+				+				
Fault-tolerance						+	+			+
Resource safety						+				
Readability								+	+	
Controllability							+	+	+	+
Navigability	+	+					+	+	+	
Simplicity					+			+	+	
Privacy							+		+	+
Security						+	+			+
Insurance						+	+			
Familiarity					+		+			
Loading time	+				+				+	+

more legible on a text-oriented Web page than a smaller size. A Web page designer may then select a larger font size in order to increase the usability (in this case, readability) of the final page layout.

6. QUIM applications and tool support

Some possible applications of the QUIM model for usability measurement and a related tool support are briefly described in this section. It was mentioned that the consolidated QUIM model is intended as a conceptual framework that:

- Gives consistent definitions of usability factors, criteria, and metrics; and
- Maps the hierarchical relations among them. It is also intended to serve as a guide in planning for usability measurement.

This planning could occur in the context of specification where the selection of broad usability factors leads to the generation of a concrete usability measurement plan. This plan would relate the selected factors and criteria to metrics and then to the types of data that

Table 5 Examples of calculable metrics in QUIM

Metric	Definition
Essential Efficiency (EE; Constantine and Lockwood, 1999)	EE = 100 × S_essential/
Essential Efficiency (EE; Constantine and Lockwood, 1999)	S_enacted S_essential = The number of user steps in the essential use case narrative,
Estimates how closely a given user interface design approximates the ideal expressed in the use case model	S_enacted = The number of steps needed to perform the use case with the user interface design (rules for counting the number of enacted steps hascome in the reference)
Layout Appropriateness (LA; Sears, 1995), Favors arrangements where visual components that are most frequently used in succession are closer together, reducing the expected time (cost) of completing a mix of tasks	LA= 100 × C_optimal/C_designed, $C = \sum P_{i,j} \times D_{i,j} \forall_{i \neq j}$ $P_{i,j}$ = Frequency of transition between visual components i and j $D_{i,j}$ = Distance between visual components i and j
Task Concordance (TC; Constantine and Lockwood, 1999), Measures how well the expected frequencies of tasks match their difficulty, favors a design where more frequent tasks easier are made easier (e.g., fewer steps)	TC = 100 × D/P P = N (N - 1)/2 N = The number of tasks being ranked, D = Discordance score, i.e., the number of pairs of tasks whose difficulties are in the right order minus those pairs whose difficulties are not in right order
Task Visibility (TV; Constantine and Lockwood, 1999),	TV = 100 × (1/S_total × $\sum V_i$)∀i S_total = Total number of enacted steps to complete the use case
The proportion of interface objects or elements necessary to complete a task that are visible to the user	V_i = Feature visibility (0 or 1) of enacted step i (i.e., how to count enacted steps and allocate a visibility value to them is defined by some rules in the reference)
Horizontal or Vertical Balance (Sears, 1995), These metrics evaluate how well balanced the screen is both vertically and horizontally (a score of 100 indicates perfect balance)	Balance = 200 × W1/(W1 + W2) W1 = Weight of side one W2 = Weight of side two Weight of a side = Number of pixels used × side's distance from the center Center = Halfway between the left edge of the left-most visual element and the right edge of the right-most element

should be collected in order to quantify the criteria. For example, a developer could within the QUIM framework devise a testing plan and benchmark reports that can be developed during the requirements phase and used later during the evaluation phase. If requirements for usability should change (e.g., an additional factor is deemed necessary), then a new usability measurement plan could be derived under QUIM. Compared with the original measurement plan, the modified measurement plan would indicate the additional data that should be collected in order to evaluate the new usability criteria. Both the original and modified usability measurement plans would have consistent definitions under QUIM, which may facilitate the integration of usability and its measurement in the software development life cycle. This goal is especially important in a software quality assurance model.
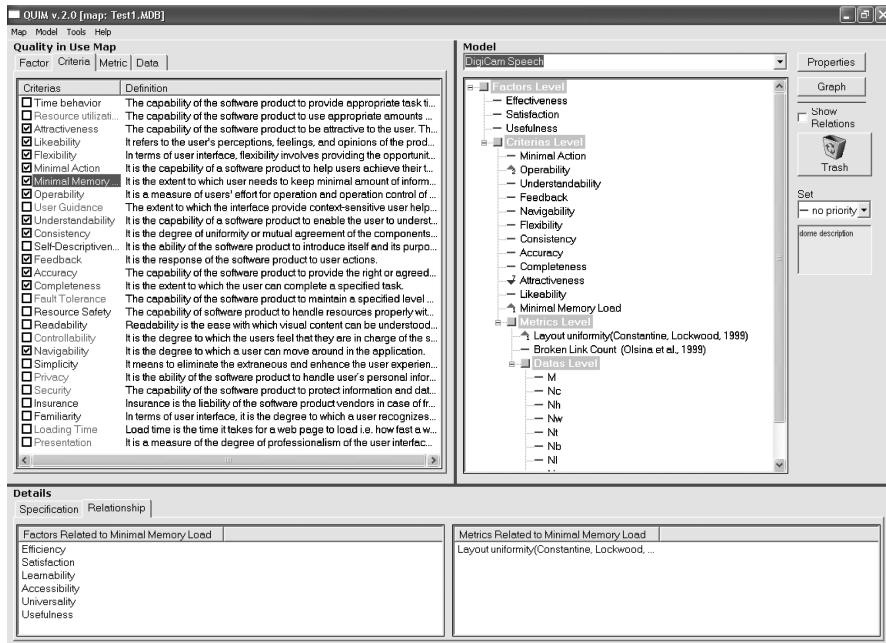
Fig. 1 QUIM Editor main interface

We developed the QUIM Editor[1] to support the kinds of activities just described and as a first step towards an integrative usability measurement toolbox for practitioners who may not be experts in usability engineering. This editor has the features that are described next:

1. The QUIM Editor supports a multi-views visual exploration of the relations among sets of factors, criteria, metrics, and data. There are also capabilities for keyword searches and goal-driven browsing that concern these relationships. Altogether these capabilities help to teach the user—developers of the QUIM Editor about usability measurement. That the QUIM Editor also suggests how to collect data required for various metrics also has pedagogical value.
2. A repository of usability measurement plans for different combinations of factors, criteria, and metrics can be created and saved with the QUIM Editor. We refer to such a repository as knowledge map for usability measurement, which could be maintained by quality assurance managers, for example.
3. There are also a few wizards in the QUIM Editor that are intended to help developers or testers who may not be already familiar with usability engineering to create a concrete, step-by-step measurement plan for a set of applications, such as efficiency measurement for safety-critical systems or accessibility in Web applications.

Presented in Figure 1 is a screen-shot of the main interface of the QUIM Editor. The editor assists the user to browse, search, and locate relevant usability factors, criteria, metrics, and data when creating a new usability measurement model or customizing an modifying an existing one. The editor also gives definitions, descriptions, and interpretations of factors,

---

[1] A free version of the QUIM editor is available at http://rana.cs.concordia.ca/odusim

criteria, and metrics; it also suggests relevant data collection methods. There is also a search function for fast navigation.

In the QUIM Editor, a new measurement plan may be created using either the "New Model" or the "Model Wizard" features. The New Model functionality allows the developer to manually create a usability testing plan by selecting the desired factors, criteria, and metrics. Such a function is suitable for usability experts who require full control over the measurement plan created for a particular application in a specified context of use. The Model Wizard functionality is intended to assist users who are not usability experts to build a usability measurement plan through a step-by-step process. Once the new measurement plan is defined, the QUIM Editor user can save it for later modification or generate a graphical map of the HTML-based report that contains all selected factors, criteria, and metrics plus information about their relations, definitions, interpretations, and calculations. This report can become part of a testing plan that is developed during the requirements specification phase. The QUIM Editor also allows the user to add their own usability factors, criteria, metrics, or specific data definitions to the program's database of such information. It is expected that only users more expert in usability evaluation may take advantage of this feature, though.

7. Conclusion

Many of the standards or conceptual models described in the beginning of this paper concern usability as aspects of software quality. Most of them are hierarchical in organization in that they decompose factors that represent hypothetical constructs about usability into measurable criteria with related specific metrics. However, the list of usability as well as the related factors, criteria, and metrics is not consistently defined across different standards or models. This make it difficult to apply them in practice because they offer relatively little advice about how to select particular metrics given broader goals of usability in a specific context of use. These problems are daunting enough for usability experts much less for those who must evaluate application usability but who lack strong backgrounds in the area of usability engineering which is the case of most developers. This motivated the need for a more consolidated usability measurement model that (1) associates factors with criteria and metrics in a clear and consistent way and (2) can be applied by usability experts and non-experts alike. That is, a usability measurement model should itself be generally usable.

Also described in this paper is our consolidated Quality in Use Integrated Measurement (QUIM) model that brings together usability factors, criteria, metrics, and data mentioned in various standards or models for software quality and defines them and their relations with one another in a consistent way. The QUIM model can help in generating usability measurement plans where specific metrics are identified and defined given various higher-level usability goals, such as efficiency, satisfaction, and learnability. These plans include information about how to collect the data needed to calculate metrics associated with the overall usability goals. The capability for multiple-perspective searching or viewing of relations among usability factors, criteria, metrics, and data is also of pedagogical value for those who are not already usability experts.

A consolidated model such as QUIM can help build a broader theory of usability measurement. That is, usability measurement must ultimately be more than a static inventory of a large number of metrics. Specifically, the collection of data to calculate various usability metrics should eventually lead to better decisions by those responsible for software development, whether those individuals are software engineers, team managers, or usability engineers. The connections between the data, the interpretations of those data, and the implications of the

results should be made in a way that is both valid and understandable to all the types of professionals just mentioned. An empirically validated model is necessary as are related metrics that provide simple and configurable questionnaire-based user interfaces that produce clear as outputs reports and recommendations about how to measure usability. More attention to the information about the user and the context is also required to facilitate the selection and the customization of many of the proposed metrics and higher-level criteria. The proposed model and its editor are intended as first step along the way toward this goal.

References

Ahuja, V. 2000. Building trust in electronic commerce, *IT Professional* 2: 61–63.

Atif, Y. 2002. Building trust in e-commerce, *IEEE Internet Computing* 6: 18–24.

Bevan, N. 1995. Measuring usability as quality of use, *Software Quality Journal* 4: 115–130.

Bevan, N. and Azuma, M. 1997. Quality in Use: Incorporating human factors into the software engineering lifecycle, Proceedings of the Third IEEE International Symposium and Forum on Software Engineering Standards, Walnut Creek, CA, pp. 169–179.

Bevan, N. and Macleod, M. 1994. Usability measurement in context, *Behavior and Information Technology* 13: 132–145.

Bevan, N. and Schoeffel, R. 2001. A proposed standard for consumer product usability, Proceedings of 1st International Conference on Universal Access in Human Computer Interaction, New Orleans, LA, pp. 557–561.

Boehm, B.W., Abts, C., Brown, W., Chulani, S, Clark, B.F., Steece, B., Brown, A.W., Chulani, S., and Abts, C. 2000. *Software Cost Estimation with COCOMO II*, New York: Prentice-Hall.

J. 1978. *Characteristics of Software Quality*, New York: North Holland.

Caldwell, B., Chisholm, W., Vanderheiden, G., and White, J. (Eds.), 2004. Web Content Accessibility Guidelines 2.0, W3C Working Draft 30 July 2004, World Wide Web Consortium. Retrieved July 3, 2005 from http://www.w3.org/TR/2004/WD-WCAG20-20040730/.

Cheskin Research and Studio Archetype/Sapient 1999. e-Commerce trust study. Retrieved June 30, 2005 from http://www.cheskin.com/docs/sites/1/report-eComm%20Trust1999.pdf.

Constantine, L.L. and Lockwood, L.A.D. 1999. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centred Design*, New York: Addison-Wesley.

Council of the European Union, 1990. Council Directive 90/270/EEC on the Minimum Safety and Health Requirements for Work with Display Screen Equipment, *Official Journal of the European Communities* L 156: 14–18.

Curtis, B., 1980. Measurement and experimentation in software engineering, *IEEE Transaction on Software Engineering* 68: 1144–1157.

Fenton, N. E., and Whitty, R., 1995. *Software Quality Assurance and Measurement: A Worldwide Perspective,* London: International Thomson Computer Press.

Friedman, B., Kahn, P.H., Jr., and Howe, D.C. 2000. Trust online, *ACM Communications*, 43: 34–40.

Hyatt, L.E. and Rosenberg, L.H. 1996. A software quality model and metrics for identifying project risks and assessing software quality. Retrieved July 3, 2005 from http://satc.gsfc.nasa.gov/support/STC_APR96/qualtiy/stc_qual.PDF.

Institute of Electrical and Electronics Engineers, 1990. 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminology*, Los Alamitos, CA: Author.

Institute of Electrical and Electronics Engineers, 1998. 1061-1998, *Standard for a Software Quality Metrics Methodology*, Los Alamitos, CA: Author.

International Electrotechnical Commission, 2004. IEC 60300-3-9, Ed. 2.0, *Dependability Management, Part 3-9: Application Guide, Risk Analysis of Technological Systems*, Geneva: Author.

International Organization for Standardization, 1998. ISO 9241-11, *Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs), Part 11: Guidance on Usability*, Geneva: Author.

International Organization for Standardization, 1999. ISO 13407:1999, *Human-Centered Design Processes for Interactive Systems*, Geneva: Author.

International Organization for Standardization/International Electrotechnical Commission, 1991. ISO/IEC 9126, *Information Technology, Software Product Evaluation, Quality Characteristics and Guidelines for their Use*, Geneva: Author.

International Organization for Standardization/International Electrotechnical Commission, 1995. ISO/IEC 12207, *Information Technology, Software Life Cycle Processes* Geneva: Author.

International Organization for Standardization/International Electrotechnical Commission, 1999. ISO/IEC 14598-1, *Information Technology, Software Product Evaluation*, Part 1: General Overview, Geneva: Author.

International Organization for Standardization/International Electrotechnical Commission, 2001. ISO/IEC 9126-1 *Standard, Software Engineering, Product Quality, Part 1: Quality Model*, Geneva: Author.

International Organization for Standardization/International Electrotechnical Commission, 2001. ISO/IEC 9126-4, *Software Engineering, Product Quality, Part 4: Quality in Use Metrics*, Geneva: Author.

Ivory, M.Y. and Hearst, M.A. 2001. The state of the art in automating usability evaluation of user interfaces, *ACM Computing Surveys* 33: 470–516.

Jarrar, M., Demey, J., and Meersman, R. 2003. On using conceptual data modeling for ontology engineering, *Journal on Data Semantics* 2800: 185–207.

John, B.E. and Kieras, D. E. 1996. Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction* 3: 287–319.

Kirakowski, J. and Corbett, M., 1993. SUMI: The Software Usability Measurement Inventory, *British Journal of Educational Technology* 24: 210–212.

Lin, H. X., Choong, Y.-Y., and Salvendy, G., 1997. A proposed index of usability: A method for comparing the relative usability of different software systems, *Behaviour and Information Technology*, 16: 267-277.

Macleod, M., 1994. Usability: Practical Methods for testing and Improvement, Proceedings of the Norwegian Computer Society Software Conference, Sandvika, Norway. Retrieved July 3, 2005 from http://www.usability.serco.com/papers/mm-us94.pdf.

Macleod, M., and Rengger, R., 1993. The development of DRUM: A software tool for video-assisted usability evaluation. Retrieved July 3, 2005 from http://www.usability.serco.com/papers/drum93.pdf

Macleod, M., Bowden, R., Bevan, N. and Curson, I., 1997. The MUSiC performance method, *Behaviour and Information Technology* 16: 279-293.

McCall, J. A., Richards, P. K., and Walters, G. F., 1977. *Factors in Software Quality*, Springfield, VA: National Technical Information Service.

Nielsen, J., 1993. *Usability Engineering*, London, UK: Academic Press.

Nunnally, J. C., and Bernstein, I. H., 1994. *Psychometric theory (3rd ed.)*, New York: McGraw-Hill.

Olsina, L., Lafuente, G., and Rossi, G., 2001. Specifying quality characteristics and attributes for websites, in S. Murugesan and Y. Deshpande (Eds.), Web Engineering, Software *Engineering and Web Application Development*, London: Springer-Verlag, pp. 266–278.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., and Carey, T. 1994. *Human Computer Interaction*, Wokingham, UK: Addison-Wesley.

Rubin, J., 1994. *Handbook of Usability Testing*, New York: John Wiley.

Schneiderman, B., 1992. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (2nd ed.)*, Reading, MA: Addison-Wesley.

Schneidewind, N. F., 1992. Methodology for validating software metrics, *IEEE Transactions on Software Engineering* 18: 410–422.

Scholtz, J. and Laskowski, S., 1998. Developing usability tools and techniques for designing and testing web sites, Proceedings of the Fourth Conference on Human Factors & the Web, Basking Ridge, NJ. Retrieved July 3, 2005 from http://www.research.att.com/conf/hfweb/proceedings/scholtz/index.html.

Sears, A., 1995. AIDE: A step toward metric-based interface development tools, Proceedings of the ACM Symposium on User Interface Software and Technology, New York: ACM Press, pp. 101–110.

Shackel, B., 1991. Usability—Context, framework, definition, design and evaluation, in B. Shackel and S. Richardson (Eds.), *Human Factors for Informatics Usability, Cambridge*, MA: University Press, pp. 21–38.

Stevens, S. S., 1959. Measurement, psychophysics, and utility, in C. W. Churchman and P. Ratoosh (Eds.), *Measurement: Definitions and Theories*, New York: John Wiley, pp.18–63.

Tilson, R., Dong, J., Martin, S., and Kieke, E., 1998. Factors and principles affecting the usability of four e-commerce sites, Proceedings of the 4th Conference on Human Factors & the Web, Basking Ridge, New Jersey. Retrieved July 3, 2005 from http://www.research.att.com/conf/hfweb/proceedings/tilson/index.html.

Yamada, S., Hong, J.-K., and Sugita, S. 1995. Development and evaluation of hypermedia for museum education: Validation of metrics, *ACM Transactions of Computer Human Interface* 12: 410–422.

Landuaer, T.K. The Trouble with Computers: Usefulness, *Usability and Productivity*, MIT Press, 1995.

Mayhew, D.J. (1999). The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface design, Morgan Kaufmann, San Francisco.

Ahmed Seffah interests are at the intersection of human-computer interaction and software engineering, with an emphasis on human-centered software engineering, empirical studies, theoretical models for quality in use measurement, as well as patterns as a vehicle for capturing and incorporating empirically valid design practices in software engineering practices. He is a co-founder of the Usability and Empirical Studies Lab and the founder and the chair of the Human-Centered Software Engineering Research Group at Concordia University.

Harkirat K. Padda is a Ph.D. candidate at Concordia University (Montreal) since 2003, where she completed her masters' degree in computer science. Her research interests are software quality measurement, metrics, and empirical software evaluation. Being a member of Human Computer Software Engineering Group, she is exploring the comprehension and usability of software systems—in particular the visualization systems. In her masters' work, she already defined a repository of different metrics to measure the 'quality in use' factors of software products in general. Currently, she is proposing a pattern-oriented measurement framework to measure comprehension of visualization systems.