The Assessment of Software Quality from the User Perspective: Evaluation of a GIS Implementation

BRONWYN HUITFELDT, Queensland Department of Natural Resources, Australia
MICHAEL MIDDLETON, Queensland University of Technology, Australia

Software quality analysis was carried out using the characteristics of functionality, reliability, usability, efficiency, maintainability and portability and their associated sub-characteristics that are defined in the national software quality standard AS/NZS 4216:1994 [ISO/IEC 9126: 1991]. The research indicates that the standard criteria are oriented towards IT maintenance staff rather than end users and are identifiable with established, stable business areas. Further criteria such as those relating to flexibility should be included for effective end user evaluation. (An extended abstract appears in: Effective Utilization and Management of Emerging Information Technologies; Proceedings of the Information Management Resources Association International Conference, Boston, May 17-20, 1998, edited by M. Khosrowpour, IRMA, Hershey, PA, USA. (ISBN 1-878289-50-0). p. 928)

INTRODUCTION

Delivering a quality software product to the end user within time and budget is the aim of all Information Technology (IT) project managers. Whether that aim is achieved is determined not only by the degree to which the system conforms to written specifications, but also by the degree of user satisfaction with the operational system.

The Aboriginal and Torres Strait Islander Land Claims System (ATSILCS) is a system that has been developed to manage the politically sensitive business area of native title matters in Queensland during a period of controversy and uncertainty associated with urgent implementation.

This paper considers the usefulness of the standard quality characteristics for evaluating quality in a system developed under such conditions. The evaluation was carried out by questionnaire and follow-up survey. The study included matching of software quality standard characteristics with survey questions.

AS/NZS 4216:1994 [ISO/IEC 9126:1991] (Standards Australia & Standards New Zealand, 1994) is a first attempt to define as an international standard, the characteristics of software quality. Although the standard defines individual quality characteristics in broad terms as sets of attributes that bear on the relevant aspect of the software to perform particular activities, it is not prescriptive, and the definitions lack clarity. Given changes in the technological and social environment, it is to be expected that this set of characteristics will be enhanced to include emerging requirements for software quality as defined by software users.

Defining quality

The IT industry has yet to produce a widely accepted descriptive scheme for assessing the quality of a software product. Definitive work has been done by researchers such as McCall et al. (1977) and Boehm et al. (1978). However, it is still difficult for an end user, or software development client, to evaluate the quality of the delivered product.

While time and cost are clearly defined concepts, quality is less easily defined (Turner, 1993). Perry (1992, p. 49) states that “meeting user requirements is a measure of quality” but agrees with Gillies (1992) that quality in an information technology context is difficult to define. A definition is
providing the International Standards Organization (1994) as “The totality of features and characteristics of a product or service that bear on the ability to satisfy specified or implied needs”.

Heidenreich (1988, p. 514) defines quality as “fitness for use or compliance with requirements”. The achievement of quality is contingent on a clear definition of function, performance objectives and design. Specifically, the project must follow explicit criteria, standards and specifications. Demming (1992) claims that this view is too limiting to enable developers to deliver software of quality and dependability, and proposes making the concerns of the customer central to the criteria for judging software.

The Denning approach has parallels with the Rapid Application Development (RAD) approach. Frequently user specifications are frozen before the technical design, coding and testing are accomplished. In other cases, users are presented with an inadequate system because the needs of the business have changed substantially during the extended time spent waiting for the system to become operational. The shorter the elapsed time between gathering user specifications and system implementation, the more likely it is that the system will be satisfactory for the users. Rapid development techniques address the need for quality by involving users in the analysis and design of the system, and address the need for speed by using CASE tools, which allow the design to be converted to code as quickly as possible.

The quality of a system developed using the RAD approach is defined as the degree to which the system meets user requirements at the time it begins operation. This is fundamentally different from the more usual definition of quality, as the degree to which a system conforms to written specifications. (University of California, Davis, 1997).

Gillies (1992) asserts that a problem with all definitions of quality is context dependence and suggests that defining quality computer software is problematical because: (a) software has no physical existence; (b) there is a lack of clear client requirements from the system at the start of development; (c) client needs change over time; (d) there are rapid changes in both hardware and software; and (e) customers have high expectations of the developed system, particularly with respect to adaptability.

Evaluating quality is equally as difficult as defining quality. One approach is to use not one single property (Gillies, 1992; Glass, 1992), but a set of characteristics against which the degree of conformity of the software to the characteristics can be evaluated. These characteristics describe the product and form the basis for the evaluation.

To take a more user-oriented approach to systems, it is instructive to consider works such as that of Taylor (1986). His concern was to establish what he called “user criteria of choice” that are generally applicable to information systems whether or not they are computer based. From this perspective, he defined “interfaces” for each criterion of choice. Therefore, a criterion such as “quality” would have interfaces such as “accuracy” and “validity” and one such as “ease of use” would have interfaces that improved “formatting” and “browsing”.

Unfortunately, Taylor’s work does not appear to have helped form the international software quality standard, but certainly Boehm’s work has. Boehm et al. (1978) established a conceptual framework and some key initial results in the analysis of the characteristics of software quality. They stated “the evaluator must be responsive to … various needs. There can be no single quality measure. The approach ... is to define a number of characteristics which span the spectrum ...”

They felt that in general when users were acquiring a software package they were primarily concerned with the questions: (a) how well (easily, reliably, efficiently) can I use it as is; (b) how easy is it to maintain, understand, modify, and retest; and (c) can I still use it if I change my environment?

They developed a set of software quality attributes (Boehm et al., 1978, p. 3-19) into a ‘characteristics tree’ showing both a hierarchy of characteristics and their interrelatedness. These characteristics are remarkably similar to the set of six quality characteristics developed sixteen years later in AS/NZS 4126:1994 [ISO/IEC 9126:1991] with portability, efficiency and reliability being common to both documents. Each of the standard characteristics is defined as “a set of attributes that bear on” the relevant aspect of software and can be refined through multiple levels of sub-characteristics (much like Boehm’s Tree).

Evaluation of the end product is undertaken by measuring how well it conforms to the characteristics: functionality, reliability, usability, efficiency, maintainability, and portability. The lack of precision in the definitions of the characteristics, and the confining of sub-characteristics to an appendix outside the official standard, detract from its usefulness as a conceptual framework (Fenton, Whitty and Lizuka, 1995, p. 13).

This has led to proposals for extended interpretation of the quality specifications. For example, van Zeist and Hendriks (1996) describe an extension to the ISO model. Using experience obtained from the QUINT (Quality in Information Technology) project, they introduced a further eight sub-characteristics to the 24 already established in the ISO appendix.

Additionally, Kitchenham et al. (1997) have tried as part of a project SQUID (Software Quality In Development) to rationalize McCall’s criteria with the ISO characteristics. They have pointed out practical difficulties such as the different ways in which the characteristics may be decomposed, the lack of standards for this, and the fact that different subsystems may have different evaluation requirements. This arises, as Dromey (1995) has pointed out, because software implies rather than manifests quality. He has proposed a model for linking quality-carrying properties to the high-level quality attributes of the ISO standard in order to deal adequately with characteristics such as user interfaces.