Journal of International Information Management

Volume 6 | Issue 1 Article 1

1997

End User Perception and Software Quality Assessment

K L. Nance
University of Alaska

M Strohmaier
University of Alaska

End User Perception and Software Quality Assessment

K. L. Nance and M. Strohmaier
University of Alaska, Fairbanks

Abstract

The topic of software quality assessment is at the forefront of the software engineering movement. Many models of organization and quality control exist which serve to foster software quality and reliability (Bloom, McPheters & Tsiang, 1973; Brandl, 1990; Comer, 1988; Dunn, 1990; Kaplan, Clark & Tang, 1994; Livson, 1988; Musa, Iannino & Okumoto, 1987). Some models now consider the software engineering project team and end users, but most still treat them as static contributors to the system. Major factors often ignored in most conventional models include the cybernetics of the process, and human factors which have a significant impact on the assessment of the quality of the software. Software environment, adaptability, and motivation have all been discussed as essential in creating a superior product. Yet, the factor of human perception has been either overlooked or avoided in the ensuing discussions. In software development the software engineers' perceptions as well as those of all potential end users, must be considered in order to ensure software quality and reliability.End User Perception and Software Quality Assessment

Introduction

A computer is a consistent machine. It will interpret the same situation in a predictable manner time after time. The interpretations of a human, however, are not consistent. They change depending on variants in conditions. The number of inputs to a computer is measurable and generally static, whereas the number of inputs to the human brain is dynamic and virtually impossible to measure comprehensively. Inputs to the human brain come from both within the individual and the external environment. The resulting differences in interpretation can result in a decline in output predictability, thus having a marked effect on the quality and reliability of software.

Cybernetics

Early Greek philosophers began to question whether individuals are born with skills, abilities, and personality, or if they developed as a result of experience. This "nature or nurture" discussion or debate has been going on ever since. Although we cannot answer the question for human beings, we are able to answer the question for computers and software. For a computer program it is all "nature." A computer program is instilled by its creator(s) with a specific logic which generally will not be changed as a result of interactions. If changes do occur, they are the result of careful programming and the changes are predictable, even if the outcome is not. One example of such a change is a feedback loop.

Cybernetics has been applied to communication and control systems in humans as well as machines. Cybernetics has been investigated primarily as a quality assurance mechanism, the fundamental basis of which includes the principle of the feedback loop. With a computer program the tendency will be toward lower entropy as opposed to the human factors, which will tend toward increased entropy. Efficient and effective software engineering results in control mechanisms that minimize the tendency toward such disorganization. Unfortunately, even the most careful software engineering cannot flawlessly predict the perceptions of the end users.

Perception

Perception refers to how an individual views input based on his or her past experience, knowledge, and current physiological and psychological states (Pearson & Spitzberg, 1990). This perception allows the individual to incorporate an appropriate response. On a simple level, perceptual inquiry investigates how identification of a particular item is possible (DeVito, 1990). On the more complex human level, perceptual research attempts to unravel such problems as how the brain translates the stationary flashing lights on a marquee into the illusion of motion, or how a person learns to interpret nonverbal communication.

An individual is continually bombarded with sensory stimuli. To allow the individual to function in a somewhat organized, less than chaotic manner, certain stimuli will be selected to be recognized and responded to, while others will be ignored (Pearson & Spitzberg, 1990). The stimuli most likely selected will be either novel, repeating, familiar, or specific to some salient need of the individual. New stimuli catch one's attention. Repeating, flashy, or brightly-colored stimuli for example, will likely be selected. Familiar stimuli are recognized because they appease the individual's desire for affirmation. Finally, stimuli that are salient to an individual's needs will be noticed; for example a thirsty person will respond to all stimuli that deal with liquids.

If a person is in a new environment, he or she will search for stimulus materials that are similar to known precepts. These similar stimuli will be used with less trepidation, as the perceptual knowledge from past experience is working as a guide. Once the new material is mastered, the individual will store the new information for future use.

Background and past experience make up an individual's perceptual base. Given that no two individuals have exactly the same background and set of past experiences, no two individuals will perceive exactly the same way. Likewise, since an individual's background is continually developing, the individual may not perceive a given situation exactly the same way when encountering it a subsequent time. Although some people do perceive more closely with one another, when considering cybernetics for a vast, heterogeneous audience, there will also be individuals who perceive in extremely different ways. Part of this difference is based in cultural variation.

Cross Cultural Implications

The increase in global marketing has resulted in a more diverse end user population for many software products. The differences in the associated perceptual bases of such target audiences can be significant. In coordinating communication between people of various cultural backgrounds, the level of heterogeneity is even higher than normal. The perceptual base between the software engineer and the end user is then, necessarily, even more divergent. When dealing with differing cultures there is the obvious potential language barrier, which can exacerbate problems between the software engineering project team and the end user. In addition, there are various, more subtle, differences that can have a major impact on software use and perception. While these differences are many, there are two general classifications that have particular relevance in a cross-culture setting.

First, cultural differences in communication are classified on a continuum between low context and high context communication (Gudykunst & Ting-Toomey, 1988). Low context communication is characterized by the use of an explicit code, to the extent that not much is expected of the person receiving the message. The message construction is both direct and succinct, and the perceived success of the interaction rests mainly on the sender of the message. The United States is an example of a culture utilizing low context communication. The U.S. even has commonly used expressions indicating the preference for the explicit form of communication, such as "get to the point," "quit beating around the bush" and "say what you mean" (Nance & Strohmaier, 1994).

On the other end of the continuum, high context communication is implicit in nature. The majority of the information resides within the physical context and/or is internalized in the person. The message construction is both indirect and diffuse. Explicit specifics about the problems and/or situation are not required. The message is manipulated around the point, clues are given toward the point so that those involved in the communication may reach it themselves (Nance & Strohmaier, 1994). The implicit nature of the communication expects more of the person receiving the message than low context communication, thus all parties of the interaction share responsibility for the success of the information sharing process. Japan is an example of a high context culture.

As alluded to previously, the directness or indirectness of communication is also an important component in the equation of cultural variability (Levine, 1985), with a finding that low context cultures utilized communication styles of directness, whereas high context cultures preferred an ambiguous communication style. Acknowledgment of the low context---high context continuum is critical, as it impacts the end user perception of computer software. As a specific example, the detail desired in the "user-friendly" aspects of programs may be affected by the directness of communication. The same program may be perceived as "too intimate" or condescending by one end user population, and "not personal enough" by another.

A final salient area of general cultural variability is chronemics. Chronemics refers to the way people use time, and is generally expressed nonverbally. The two basic types of time use across cultures are referred to as monochronic time and polychronic time (Hall, 1983). Monochronic time refers to doing one thing at a time, with a linear view of time. Polychronic time is characterized by doing several things at once, with a more "circular" or ongoing perception of time. As an example, arriving at a meeting late would be more traumatic to an individual using monochronic time, as a polychronic culture

member would be much more liberal with the definition of what was considered "on time." This difference in perception of time and time use may have implications on end user perception as well (Nance & Strohmaier, 1994)

Perceptual Issues

A cost-benefit analysis approach is widely used to determine the appropriateness of an item in many applications. In determining the cost-benefit ratio for any computer program, there are many determinants which can be classified as intangible. When attempting to pre-determine the needs and wants of potential end users, the programmer must attempt to weigh these intangibles. Stemming from the discussion of perception, any two people will likely assign different importance to intangibles based on their desires, past experiences, and personal likes and dislikes. Some of the intangibles that may be subject to cost-benefit analysis include timing delays, security, system reliability, and data reliability and integrity.

Timing delays

The perception of the software engineering project team as to what constitutes an "acceptable delay" may differ significantly from the perception of the end user as to what constitutes an acceptable delay. The experience of the software engineer enables him or her to have the breadth of knowledge sufficient to recognize what goals are realizable with respect to timing delays. In addition, the software engineer has the knowledge necessary to make value judgments as to the intangible costs associated with various potential solutions to a user-provided problem. The expectations of the user may be unrealizable, or far more costly in terms of how the speed-ups may affect other system components.

The question then becomes "what constitutes a long delay?" From the user's perspective, a long delay may constitute a specific period of time, or merely a period of time during which stimuli are not presented. Even though the actual time delay remains constant, users may perceive a tangible difference between a) waiting while staring at a blank screen, b) waiting while staring at an unchanging screen, and c) waiting while staring at a changing screen. The levels of importance assigned to these differences will vary with each individual user, and user culture, as will the perception of these differences as being more or less important to them. Since the product or system may be utilized by more than one individual, an important part of software quality assessment is the impact that the timing delays will have on the majority of the users, as well as the individual users.

For example, a "hunt-and-peck" typist in an office will be satisfied with a basic word processing program, and probably not be too concerned with the throughput capabilities of the application software. However, a person in the same office with advanced input capabilities would likely not be satisfied with the same basic program. Waiting for periodic backup will also frustrate some users. Another example of user perceptions differing from that of the software engineers can be seen in early versions of several word processing programs. Frustration levels were high for fast typists as the programs frequently had to "catch up" to the typists.

As for cross cultural ramifications, what sort of implications exist when a monochronic engineer is designing and/or implementing a program that will be used by a polychronic receiver, or vice-versa? In determining end user perceptions, the effect of chronemics must also be considered in order to produce an efficient and effective quality software product.

Security

Security refers to the ability of non-authorized persons to access components of the program. These non-authorized persons may be either external or internal to the organization. Whereas internal security refers to the security of code from personnel within the organization, external security is protection from sabotage and foreign meddling. Again, the level of acceptable security differs with each user group. An additional factor that is unknown to the software engineering project team is the level of personal integrity of the individuals within the organization. Often the stereotype is either a) the individuals are trustworthy and moral, or b) those with access to the program are liable to contaminate or embezzle.

There exist cultures and individuals within cultures who will obey laws primarily because they are laws. Others require active enforcement or sanctions in order to respect a law, policy or ordinance. The role that culture plays in the reaction of an individual to laws or ordinances is very significant (Nance & Strohmaier, 1995). The stereotype that the software engineer holds, and that which the user believes, both impact the perception of what degree of internal and external security is acceptable and necessary to ensure software quality.

System Reliability

The reliability associated with a particular application will generally differ significantly between the user and the software engineer. A user may consider a program to be robust and reliable if it works well, and consistently, on his or her particular system. From the perspective of a software engineering project team, the application program may not be considered either robust or reliable unless it is capable of being executed in a variety of user environments. Therefore, a program which is deemed reliable by one user group may be perceived as unreliable by another user group. There may be no actual difference in the applications, but rather a differential associated with the platforms on which the applications are run.

Technological changes which may occur in the near future are difficult to anticipate, as are the effects these changes will have on assessing the reliability of a particular application. In addition, there are organizational changes such as internal diversification which may alter the platforms on which a particular organization utilizes a particular application. Thus, perception is an important component in measuring software reliability, and accurately determining software quality.

Another issue which may result in perceptual differences is the responsibility of the software engineering project team to ensure that effects of hardware problems minimally affect the integrity of the overall system. Many software engineering project teams will consider this a hardware problem and will not spend a large amount of programming effort writing error code which will most likely never be

referenced. The user's perception may differ, with much more weight being given to the importance of this rarely referenced code. This issue is more tangible than most of the others presented herein, as the person-hour assessment of the cost of building in extra checks can readily be assessed and presented to the user for an accurate cost-benefit analysis.

Data Reliability and Integrity

The user of a system desires data reliability and integrity. If an individual enters a particular piece of data, it is expected that the data will be available in the future for use by the individual and possibly others. Difficulties which may arise relating to perception in data integrity include frequency of updates and the conversion of data to information. Frequency of updates has been addressed briefly under timing delays, as an update generally causes some sort of timing delay.

One question frequently asked of users is how much information they would be willing to re-input to prevent unnecessary timing delays. Unfortunately, this question does not guarantee consistent amounts of data. Since updates are generally performed according to some internal clock, the amount of data entered frequently corresponds to the speed of the user rather than to the time. While ten minutes may be enough time for one user to enter only a few records, another user may enter hundreds of records in the same time period. The software engineering project team may help alleviate this problem by considering potential user differences and allowing users to control update frequency.

Another difficulty which may arise in this situation is initiated by the subtle difference between data and information. Data is in raw form. It does not become information until it is organized into some meaningful order. Data is what is stored in the computer, and what the software engineering project team is generally dealing with. A difficulty may arise which subtly shifts the traditional definition of data integrity. Will the data, when organized as information, be perceived by the user in the manner in which it was intended? Perceptual differences by individual users or user groups may lead to different informational interpretations of the same data. The software engineering project team can minimize this by taking the potential perceptions of the users into consideration and organizing data in such a manner as to minimize potential differences in interpretation.

Although perceptual consideration is not traditionally a role of any member of the software engineering team, it is a traditional component in the field of information science. Information science is a relatively new field which attempts to tangibly analyze information, including its creation, transmission, transformation, encoding, measurement, use and valuation. This last area of information science, valuation, is the subfield which most closely associates with perception.

Information scientists attempt to scientifically analyze anything which affects any aspect of information. The biggest difference in information science and the traditional software engineering project team is the extent to which they consider human thought processes. Information scientists use perception as a major foundation in attempting to create information systems which are easy to understand and use. Software engineering project teams can help ensure this new type of data reliability and integrity by taking into consideration many of the factors emphasized in the field of information science.

Summary

Quality assessment measures are evolving for software engineering projects in an attempt to both produce a superior product, and to measure the quality of a product. Unfortunately, the measures do not always yield consistent results, largely because the human perception aspect has been either overlooked or avoided. In software development the software engineer's perception must be considered in order to ensure software quality and reliability. In addition, all potential end user perceptions should be taken into consideration. These perceptual differences include both intracultural and cross cultural foundations (Nance & Strohmaier, 1995), given the world-wide marketing of products.

Software quality assessment instruments must be developed and modified to account for and incorporate the perceptions of the individuals associated with the process. Modifications should be weighted to predict and account for the factor of human perception which potentially affects each particular system. Once consideration of the human factor has been effectively incorporated into the software quality assessment process, accurate and consistent quality assessments can be expected.

References

Bloom, S., McPheters, M., & Tsiang, S. (1973). ***Software Quality Control. IEEE Symposium on Computer Software Reliability***. New York: IEEE.

Brandl, D. (1990). Quality measures in design. ***Software Engineering Notes, 15.*** ACM Press.

Comer, E. (1988). Software quality engineering: Making the myth a reality. ***Proceedings of the Annual National Joint Conference on Software Quality and Reliability***. Washington: NSIA.

DeVito, J. (1990). ***Human communication: The basic course***. New York: Harper Collins.

Dunn, R. H. (1990). ***Software quality: Concepts and plans***. Englewood Cliffs, NJ: Prentice-Hall.

Gudykunst, W. B., & Ting-Toomey, S. (1988). ***Culture and interpersonal communication***. Newbury Park: Sage.

Hall, E. (1983). ***The dance of life***. New York: Doubleday.

Kaplan, C., Clark, R., & Tang, V. (1994). ***Secrets of software quality***. New York: McGraw-Hill.

Levine, D. (1985). ***The flight from ambiguity***. Chicago: University of Chicago Press.

Livson, B. (1988). A practical approach to software quality assurance. ***Software Engineering Notes***, 13. ACM Press.

Musa, J., Iannino, A., & Okumoto, K. (1987). ***Software reliability: Measurements, prediction, application***. New York: McGraw-Hill.

Nance, K., & Strohmaier, M. (1994). Ethical accountability in the cyberspace. ***Proceedings of the Ethics in the Computer Age Conference***, 115-118.

Nance, K., & Strohmaier, M. (1995). Ethical information security in a cross-cultural environment (pp. 603-11). In J. H. P. Eloff & S. H. von Solms (Eds.), ***Information security - The next decade***. London: Chapman and Hall.

Pearson, J., & Spitzberg, B, (1990). ***Interpersonal communication: Concepts, components, and contexts***. Dubuque, IA: Wm. C. Brown.