

The Dimensions of Software Quality

Michael W. Usrey
Protocol Communications, Inc.
Minneapolis, MN 55432

Kevin J. Dooley
Arizona State University

Cite: Usrey, M., and K. Dooley (1996), "The Dimensions of Software Quality," *Quality Management Journal*, 3(3): 67-86.

ABSTRACT

Theoretical developments in software quality could be greatly enhanced if a theoretical definition of software quality existed. Content analysis is used here to objectively identify the dimensions of software quality. The dimensions are derived from the perspectives of both producers and consumers. Results show that the most important software quality dimensions include accuracy, capability, communication, completeness, conformance, features, flexibility, serviceability, simplicity, stability, and structuredness. Consumer and producer attitudes about software quality differ only slightly. Further analysis suggests some possible meta-dimensions of software quality: fulfilling customer requirements, ease of use, and capability (for producers of software); and fulfilling customer requirements, ease of use, and first impression quality (for consumers of software). Consumers appear to be primarily interested in what the software can do for them now, whereas producers appear to be primarily interested in what the software can do for them in the future. We shall detail the steps involved in the identification of significant dimensions, and discuss implications to practice and further research.

key words: software quality, content analysis, quality dimensions

1 Introduction

The importance of software in our lives is growing daily. People's personal and professional lives can be greatly enhanced by the presence of high quality software, and can be greatly imposed upon by the presence of poor quality software. Our most complex systems, such as airplane flight control or nuclear power plants, depend critically upon the reliability of potentially fragile software. Software is no longer limited to computers either--most consumer appliances, including communication devices and automobiles, have significant software components. The need to understand, control, and design quality software is unprecedented [Gibbs, 1994].

This work is based on the premise that a deeper, more objectively-based understanding of the *components*, or dimensions of software quality would be of benefit to both practitioners and researchers. Such a list of dimensions would help practitioners understand the nature of software quality and could facilitate design and market decisions; from a theoretical perspective, sound definition increases the efficiency and effectiveness of subsequent research in the area [Kuhn, 1970]. Our work is motivated by similar research done by Zeithaml, Parasuraman, and Berry

[1990] which helped identify the dimensions of service quality, and by the seminal work of Garvin [1988] which outlined the dimensions of product quality.

Applications of a formal definition could influence software engineering practice in several ways. Explicit quality criteria could be more easily identified and incorporated into the software development waterfall verification and validation process. Dimensions identified in the research might be used to enhance existing methodologies such as Quality Function Deployment or Basili and Weiss' [1984] method for software engineering experimentation. The research could also be used as justification to invest in new software development tools and techniques. New theory would introduce new insights to characterize software and the quality response. A new model for software quality should have a favorable impact on software metrics research. The research should also suggest practical priorities in software management.

The research to be presented addresses the following:

1. A shortcoming of existing software quality theory and a unique feature of this research pertains to the use of attitude measurement theory articulation techniques and the unique application of these techniques to software quality research; the research provides a model for future empirical attitude research in software product quality.
2. The bulk of existing software quality research is concerned with the processes of software producers. Little attention has been focused on software consumers. That shortcoming was addressed by regarding the software consumer as an integral research element.
3. New theory should efficiently model known phenomena and facilitate novel predictions. Every effort was made to discover unique characteristics that impact the software quality response.

A review of previous definitions of (software) quality is presented. Subsequently we shall describe the manner in which we empirically identified significant quality dimensions, discuss the implications of the findings, and suggest future research avenues.

2 Definitions of Quality

Definitions of Quality via Software Engineering

There are five disciplinary approaches widely used in quality definition [Garvin, 1988]:

- Transcendent--excellence invariant over time, cost, and requirements
- Product-Based--a function of product attributes
- User-based--user's perception of system performance
- Manufacturing-Based--compliance with formal product specifications
- Value-Based--ratio of performance to cost.

Each of these views has been applied to the analysis of software quality.

In software, the Transcendental definition is similar to the programming in-the-small concept of "elegance". Guaspari [1985] and Pirsig [1974] define the Transcendental quality school: even though Quality cannot be defined, you know it when you see it. The principal problem with this approach is its lack of rigor. Only through repeated observation of artifacts that possess quality can an individual abstract, internalize, and be trained to recognize quality, and such experiences are intensely personal.

The Product-Based definition is illustrated by Fisher and Light [1979] who define software quality as the composite of all attributes which describe the degree of excellence of the computer system. Reifer [1985; as cited in Schulmeyer, 1987] and the ISO 8402 standard [1990] state that software quality is the degree to which a software product possesses a specified set of attributes necessary to fulfill a stated purpose. Hopcroft and Ullman [1979], and Hofri [1987] use automata theory and computational complexity theory to define software quality in terms of two properties: memory consumption and computational time. McCall [1977] defines numerous dimensions of software quality at different levels of abstraction. Boehm et al. [1978] assert that quality and utility are synonymous and then illustrate a stratified decomposition of utility. At the lowest level

of decomposition are phenomena that were believed to be manageable via metrics. The main focus of the work is anomaly detection.

The Product-Based definition has the appeal of rigor compared with a Transcendental definition. However, it is questionable whether adding attributes means the creation of more quality or the creation of a new and different product. There also remains the fundamental question of which attributes are most significant to software quality.

In the User-Based definition, quality is in the eye of the beholder. Prell and Sheng [1984] define software quality as the users' perception of the system's performance. The IEEE *Standard Glossary of Software Engineering Terminology IEEE-STD-729-1983* [1983] and Ramamoorthy et al. [1984] define quality as the degree to which a customer or user perceives that software meets his or her composite expectations. This approach carries the implication that software quality is not an absolute, but can vary with time and consumer expectations. The User-Based model illustrates the importance of quality as a marketing activity prior to software development.

The Manufacturing-Based approach to software quality is illustrated by numerous works which have focused on how to provide quality in the development process [Albrecht, 1979; Boehm, 1984; Cho, 1987; Curtis et al., 1979; Department of Defense, 1985; Dreger, 1989; Fagan, 1976; Humphrey and Sweet, 1987; Jones, 1990; Kemerer, 1987; Schulmeyer, 1987]. Software inspection is used in a fashion similar to manufacturing inspection to assure compliance. In both applications, inspection suffers from diminishing returns from increases in inspection resources. Measuring the quality of the development process is accomplished through the use of productivity metrics [Halstead, 1979]. Metrics, however, have not produced a great number of converts in the ranks of software management [Symons, 1991], and are often unpopular with software developers. Existing metrics tend to be highly dependent upon code length, language, compiler technology, hardware environment, and target application. These factors may explain how, even

with growing pressure to manage increasingly expensive software development, the number of individuals performing research in quality metrics is decreasing [Shen, 1987]. This is another indicator of diminishing scientific returns for the extension of this paradigm.

The Value-Based definition implies that elimination or diversion of non-value-added activity to value-added activity is synonymous with quality [Crosby, 1979]. The business rewards of quality (value) are increased profitability and competitiveness. The concept of software engineering consistency, freedom from contradiction [Boehm, 1984], and Quality Function Deployment are examples of the value-based metaphor. The concept of value combines elements of product attribute selection, specification conformance, cost, and customer perception. This overlap with the other quality models makes the concept of Value-Based quality only slightly more concrete than the unqualified term "quality". Another drawback to this paradigm is that the feasibility of measuring the composite value of an attribute-laden software system is questionable.

Software quality is an overloaded, complex concept related to attitudes. Individually, the five disciplinary approaches may contain the necessary elements for a theory of software quality, but none is sufficient by itself. As a group, there are contradictions and overlaps between the individual models.

Definitions of Quality via Quality Strategy

Garvin [1988] recognized the shortcomings of the five disciplinary quality models when he proposed his eight dimensions of quality. The dimensions are intended to describe a practically exclusive and exhaustive set of constructs that form a basis for product quality assessment. Unfortunately, Garvin's dimensions reflect a focus on fairly simple products with small attribute sets.

The complexity and large number of attributes associated with many software systems challenges the applicability of Garvin's model of product quality. Most software systems rival a manufacturing system in their complexity. The software instructions can be viewed as a production process where the result is a particular calculation, report, graphic rendering or similar response [Cho, 1987]. Software designs tend to have fewer redundant elements, i.e., more complexity, than most mechanical, electrical or architectural designs. Software, like a process, is also more likely than most physical artifacts to be used for an unintended purpose. This suggests that robustness and adaptability are keys for the long-term survival of a software system. Manufacturing Resource Planning and expert systems are examples of software that are both product and complex process.

Misterek, Anderson, and Dooley [1990] defined dimensions of process quality. Processes are more complex than products because they contain fewer redundant elements and are expected to adapt to a more varied set of requirements. The eleven dimensions of process quality are measures of the underlying systems for development, manufacture and distribution of goods and services.

Zeithaml, Parasuraman, and Berry [1990] identified the need for a different framework for the evaluation of service quality. They observed that services differ from goods in three important ways: (a) services are intangible, (b) services are relatively more heterogeneous than products, and (c) production and consumption of services are virtually simultaneous. Software certainly can be described by items (a) and (b); if one assumes software also generates information [Cho, 1987], then item (c) is also applicable. The ten dimensions of service quality led to development of the SERVQUAL instrument for measuring service quality.

The preceding discussion clearly indicates that a paradigm for software product quality does not exist within software engineering or other academic disciplines. It appears as though software can

be described simultaneously as a product, a process, and a service. The lack of paradigm is illustrated by the fact that none of the existing theories has attracted the critical mass of adherents required to drive continued development of the discipline and by the diminishing returns for additional research invested in current software quality methodology.

3 Theory Articulation

3.1 Initial Data and Assumptions

In the early stages of theory generation, retrospective analysis of existing data is a relatively low cost way to refine a theory. Manifest content analysis --the structured analysis of textual documents -- was used to generate the theoretical constructs. This technique is inherently broad and yields reliable results [Holsti, 1968]. Initial nominal definitions for each of the current theory constructs are provided in Table 1. These constructs and their definitions are drawn primarily from the references cited in the previous section. Boehm et al. [1978] was considered to be a seminal existing work regarding software quality; Garvin [1988], Zeithaml et al. [1990], and Misterek et al. [1990] were considered the works which best addressed the dimensions of quality from product, service, and process perspectives. We should note that, although it was not part of the original source literature, our proposed dimensions were found to overlap with the software characteristics identified in the ISO/IEC 9126 [1991] standard for software quality evaluation: conformance (ISO's "functionality"), durability (reliability), access (usability), efficiency, serviceability (maintainability), and independence (portability).

The ethnographic distinction between software consumers and producers was considered relevant to the response and a key distinguishing factor for this research effort. These model elements make feasible the deduction of simple hypotheses such as "Consumers place relatively more significance on product Safety than Producers."

Consideration of software delivery media, documentation, and other physical artifacts was not included in the research. Such items weren't seen as essential to the software itself and though such items are of great concern during initial deployment of software systems, they usually decrease in importance once the product has achieved its typical operating status. Concepts related to the organization selling or supporting the software were also omitted from the research. Vendor process capability, financial strength, and other important parameters are certainly a concern as suppliers become increasingly like business partners. Many of these issues have a broad base of published research and all seem to offer fertile ground for further study. In this study, recurring software operation was generally emphasized over non-recurring tasks like installation and initial training.

3.2 Publication Selection

Published works formed the unit of analysis for the experiment. The publications considered for content analysis represent attitudes towards software products and product development. Documents were classified as representing the views of either software producers or software consumers. Analysis was limited to English-language publications. No conscious attempt was made to limit publications by author or publication reputation. Several examples of documents representing software producers were identified. Papers from technical proceedings appeared to be representative of the software producer.

To be included in the sample space, titles of papers from proceedings had to include both the words "software" and "quality". Papers fitting this description were identified from the INSPEC database which is published by the Institution of Electrical Engineers. The on-line version of INSPEC is limited to papers published from 1988 to 1992, but includes over 5000 journals and proceedings. If more than one paper in a single bound proceedings met the criteria for inclusion, both were included. The authors randomly encountered additional papers from proceedings with titles that fit the criteria for inclusion.

Several other types of published material were considered and rejected. An example is in popular periodicals, where software product advertisements reflect either the attitudes of software producers, or what they want consumers to believe are their attitudes. However, many advertisements depend on visual images to create an impression on the consumer. Such advertisements are a form of latent content that is difficult to analyze with reliable manifest methods. Therefore, advertisements were omitted from the research. Articles posted to the "comp.software-eng" news group on *usenet* are also likely to be representative of software producers. However, this source suffers from a bias towards the institutions most likely to have access to *usenet*..

There are far fewer publications that are clearly and broadly representative the software consumer. Government specifications for software development, like DOD-STD-2167 [1985], are an influential source of software consumer expectations. However, these standards focus on the process of developing or procuring software, not aspects of the software itself. The impact of government standards on software quality is currently an area of active study. Software user groups' news letters were considered as a source of consumer attitudes. However, these groups tend to focus narrowly on a single software product and a constrained application domain. Text referring to the business of the organization may dominate the periodical, introducing validity concerns in the research. The organization of newsletters is also more irregular which makes consistent sampling a concern. For these reasons, software user-group news letters were not included in the research. In practice, a researcher focusing on specific market niches could include this sort of periodical to improve the validity their sample.

Software consumers write letters to the editors of popular periodicals, but this potential source of consumer attitudes was excluded from research consideration. Producers also write in response

to previously published statements and do not always disclose their company affiliations. The result is bias in either the letter author or in the research trying to cull inappropriate letters.

In product reviews, contributing authors assume the role of consumer. The reviewer may have no direct involvement with the development of software, but still may not reflect all the concerns of consumers in a given market niche. This research was concerned with software-only products, so there was also some level of expertise required on the part of the researcher to select reviews of only those products that were relevant to the research. Though not ideal, product reviews were adopted as the source of software consumer attitudes and concerns. Popular software periodicals included in the sample space were those encountered by the authors at the 1992 *Fall Comdex* trade show, at various Twin Cities computer retail outlets and encountered randomly.

To increase validity, analysis units consisting of entire works were adopted. A whole paper and all of the text in a product review were considered to be more valid indicators of publication content than samples. This resulted in word volumes that could vary between analysis units. This also suggested analysis units with much greater word volume than typically encountered. A total of 339 papers from proceedings are included in the sample frame for software producers; 58 popular periodicals represent consumers. The process of selecting specific publications for analysis is described: Randomly select a publication; computer programs and random number tables may be used for this purpose. Proceedings and product reviews out of print or otherwise unavailable are removed from the sample space and an alternate selection is made. For papers from proceedings, sampling is performed without replacement. After one of these publications is analyzed, it is dropped and the sample frame is reduced by one. For product reviews, a popular periodical is selected and the most recently published issue is used. If the periodical is no longer being published, the title is dropped from the sample frame. If the periodical contains no software-only product reviews, another sample is selected. All product reviews greater than one

page in length within a selected popular periodical are included in the sample space. The forty-eight publications selected and analyzed are identified and described in Appendix A.

3.3 Data Collection

The mechanics of the data collection for each publication depend upon the publication type. For papers from conference proceedings, all pages were analyzed, including headings, captions of figures, illustrations and bibliographies. All product reviews in popular periodicals that met the criteria were included in the analysis. The entire lexical content of the article is analyzed including title, text, captions, graph axes, tables, bibliographies, and author biographies. A sample of the written instructions for the individual tabulating the content analysis of product reviews in popular periodicals is included in Appendix B.

To conduct manifest content analysis of the documents, a coding scheme was developed that ties key words or "symbols" in the publication to various theory constructs. Symbols can be positive or negative examples of the construct. Babbie [1992] calls these symbols construct indicators. Construct indicators are used in manifest analysis to increase response frequency. Symbol frequency is the unit of observation for the experiment. The coding scheme is detailed in Appendix C.

Construction of a symbol coding scheme has a significant impact on research validity. The number of symbols per research construct was fixed at four. A small number of symbols per construct increased the likelihood of a valid mapping while reducing the opportunity for omission and error during data collection. It was relatively easy to find at least four symbols per construct, but some constructs suggested an even larger list of potential key words.

The key words representing each construct were collected in several ways. Initially, key words were identified from the original reference from which the construct definition was drawn to

enhance face validity. When required, the symbol mapping was extended with thesaurus reference [Morehead and Morehead, 1978]. Finally, additional symbols were identified during the pilot application of the instrument.

Tabulation of symbol variants was adopted to increase the response frequency. Therefore, the individual performing the word tally was responsible for identifying valid affixes applied to key words. The increase in response frequency afforded by inclusion of such variants was felt to mitigate the small reliability penalties.

Because of the variety of academic disciplines represented in the construct definitions and the breadth of software products considered, key words were potentially overloaded. For instance, the symbol "sound" would usually be associated with the Stability construct, like the phrase "sound as a dollar," but specific pieces of software or software development procedures may incorporate sound generation, editing, and playback as Features. Both meanings could be used within the same publication. Overloading of key words was most acute when mapping to the Features construct. This forced some sacrifice of research reliability to assure validity.

Another issue involves the use of a product and/or methodology proper name. This introduces potential for bias because the proper name is likely to be used repeatedly. It was decided to include each instance of these repeated words because it was assumed that the use of those words in the proper name implied the desire to create a strong association with the implied content. Cursory investigation of automating the tallying process suggested little benefit. Therefore human reviewers were used under the following guidelines: (a) a single reviewer was used for each publication, (b) publications were analyzed in roughly the same physical environment, and (c) the analysis of a given publication was completed in one session. The repetitive nature of the analysis task dictates brief rest intervals between publications for the reviewer. The order of publication review was fully randomized. Finally, it should be stressed that tabulation errors almost certainly

occurred. This fact will have some negative impact on research reliability. However, human errors should be randomly distributed among publication types, key words, and theory constructs. A test of inter-rater reliability used one consumer publication and one producer publication; it showed no statistically significant differences in the final tallies of construct frequency (at 95% confidence).

The symbol tallying process is best described by example. The following passage is excerpted from "Control Data Corporation's Government Systems Group Standard Software Quality Program" by Redig and Swanson:

One embarrassing side effect of the existing plans was the inconsistency that developed between the different GSG divisions. This inconsistency was quite embarrassing when the Defense Contract Administration Services (DCAS) representatives came to audit the SQE programs. DCAS found many different ways of doing the exact same thing. Because each program was free to develop their own plan, each program had a different set of procedures that they followed in implementing the plan. However, this left the SQE analyst trying to explain why their set of procedures were better than plans which DCAS representatives had reviewed earlier. The SQE analysts were confuse because they had procedures that were different across the GSG divisions; further the DCAS representatives were confused because they had to learn a whole new set of procedures each time they audited a different program.

Thus, the necessity for a standard SQP was clearly established. The functions of a standard SQP include a central SQE resource for supporting software proposals and bids, ensuring compliance to a common plan, training new SQE analysts, providing a consistent SQE reference source, gathering SQE metrics, performing SQE trend analysis, implementing new SQE technology, coordinating SQE technology and issues between divisions, and maintaining the current SQP documentation.

The Consistency construct is indicated twice in the first paragraph and once in the second paragraph by variants of the word "consistent". Use of the symbol "standard" twice in the second paragraph implies the Stability construct. Capacity is suggested by the symbol "resource". The key word "coordinating" is assumed to indicate the Synchronization construct. The presence of the symbol "maintaining" suggest the Serviceability construct. Finally, the word "clearly" is captured as evidence of the Legibility construct. This last mapping illustrates the trade-off between reliability and validity in a strictly manifest analysis.

The unit of analysis is an article and application of the tallying process to a single consumer or producer article generates a vector of construct frequency counts expressed as non-negative integers. A small number of the publications from those identified for the publication analysis phase were used to perform a pilot evaluation of the key word mapping scheme. Pilot evaluation of research instruments enhances research validity. If needed, additional symbols were identified at this point. This also provided an opportunity to add the inductive analysis of applying the scheme to the deductive reasoning of creating of the scheme. Documents used in the pilot iteration were discarded from the sample frame to avoid bias. Tally counts were performed by the two authors and another individual independently; discrepancies in final counts were resolved through consensus.

4 Results

Taking the consumer article tallying vectors as rows and the proposed quality constructs as columns forms a 33x25 contingency table of two nominal variables with joint probabilities π_{ij} . A similar 15X25 table can be expressed for the producer articles and the proposed quality constructs. Chi-squared tests of the null hypothesis of independence between construct and article response were conducted for both producer and consumer data. The null hypothesis was rejected in both cases with significance levels approaching zero using 336 and 800 degrees-of-freedom, respectively.

From a producer perspective, the most frequent quality dimensions found were: Accuracy, Capability, Completeness, Conformance, Flexibility, Serviceability, Stability, and Structuredness. From a consumer perspective, the most frequent quality dimensions found were:

Capability, Communication, Completeness, Conformance, Features, Flexibility, Simplicity, Stability (Table 2).

Screening techniques were applied to find the most important explanatory variables for the response pattern identified by the contingency table test. Lohlin [1992] describes principle component techniques for identifying important factors among a large number of latent variables. In standard principle components analysis, a $J \times J$ diagonal matrix is generated whose elements, r_{mn} , are the Pearson correlation coefficients for columns m and n in each $I \times J$ contingency tables above. This correlation matrix has 1's on the diagonal and all matrix elements have values between -1 and 1, inclusive. Eigenvalues are then obtained for the correlation matrix. In Kaiser-Guttman analysis, all factors with an eigenvalue greater than one are assumed to be non-trivial. A graphical variation on this technique is the scree test. First, a pareto graph of the factor eigenvalues is plotted. The plot is examined to determine the point at which the curve flattens. The factors to the left of that point, with large decreases between factors, are considered nontrivial.

Within the producer articles there are six factors identified using the Kaiser-Guttman threshold. Using the scree criteria, one factor appears to have a significant response in the producer sample. The Kaiser-Guttman test also generated six factors when applied to the consumer article data. One of the six factors was marginal with an eigenvalue near unity. Scree analysis suggests that the one dimension was significant among consumers as well. Since the Kaiser-Guttman and scree criteria produced a different number of significant dimensions, a compromise was used and the first three principle components were evaluated for both the producer and consumer article data. These three principle components account for 71% of the variation in the producer data and 69% of the variation in the consumer data. A fourth factor in both cases would contribute less than 10% additional variance coverage. Eigenvectors were used to determine the relationship between the principle components and the original dimensions that formed the basis for the correlation

matrix. Original dimensions with eigenvector coefficients with absolute values greater than .25 were considered in the interpretation. The original dimensions that accounted for less than two percent of the total response were eliminated from analysis.

Within the producer articles, Conformance, Completeness and Accuracy all made significant contributions to the first principle component (Table 3). Conformance also had the highest frequency among producer articles, accounting for over one third of the response. Capability, Accuracy, Flexibility and Descriptiveness were significant elements of the second producer principle component. The dimensions Serviceability, Simplicity and Communications correspond to the third principle component for producers. Structuredness, the second highest frequency response among producer articles, and Stability, the response with the fourth highest frequency, failed to make a significant contribution to any of the first three principle components.

For the consumer articles, Features, Conformance, Completeness and Capacity were significant elements of the first principle component (Table 4). Features was also the highest frequency response among the consumer articles, accounting for over 15% of the response.

Communications, Simplicity, Structuredness and Efficiency made significant contributions to the second principle component. Simplicity, Stability and Perceived Quality dimensions correspond to the third principle component. Capability and Flexibility, each accounting for roughly 10% of the total consumer response, were not strongly associated with any of the first three principle components.

5 Discussion

There appears to be considerable similarity between the consumer and producer article responses. The first principle components for both consumer and producer data included both Conformance and Completeness among their significant components. These dimensions also had a relatively high frequency response in both sets of data. Taken together, Conformance and Completeness

are equivalent to Boehm's compound characteristic "Completeness" and relate to the fulfillment of explicitly stated requirements. Some of his operational definitions can be applied to the constructs as defined in this treatise and others should be relatively straight forward to develop. These results support construct validity and demonstrate some longitudinal stability for those constructs.

The Features construct had the highest frequency response among consumer articles and was also a significant element of the first principle component for the consumer data. This result is consistent with the results obtained by Garvin in *Managing Quality* which supports concept validity. As defined during manifest content analysis, the Features dimension is the combination of Garvin's Performance and Features categories. Garvin discriminated between primary product operating characteristics and secondary operating characteristics. Software products have numerous attributes which makes partitioning between primary, secondary and even more classes of operating characteristics difficult and somewhat arbitrary, so the two dimensions were combined for this analysis. Garvin's dimensions are subtly different from Boehm's "Completeness" in that they include both stated and unstated requirements. Within established software quality practice, software development is concerned only with those characteristics necessary to satisfy the stated requirements, there is no stratification of stated requirements, and unstated requirements must be made explicit [McManus, 1987].

The distinction between, and relative importance of, explicit versus implicit requirements could also be indicative of a nominal variable characterizing make-to-stock versus make-to-order software. Though the concept is common to manufacturing disciplines and seems to be a plausible way to characterize software products, the distinction appears rarely, if at all, in software quality and software engineering literature. Early software development was often concerned with make-to-order systems [Brooks, 1975; Boehm, 1978], so software engineering standards and practice would reasonably have evolved from that foundation.

However, that assumption may well have to be reassessed given the increasing importance of consumer software that is essentially make-to-stock.

All the consumer product reviews in the original sample are clearly associated with make-to-stock software. The majority of producer articles appear to be concerned with make-to-order software while the other papers cannot be exclusively categorized as either make-to-stock or make-to-order. Therefore, the stock-order variable is confounded with the consumer-producer variable and cannot be controlled or blocked in the data. Either variable could account for the difference in response.

Accuracy, the correctness of a program, was significant in both the first and second producer principle component. This dimension could be an expression of a baseline expectation for software developers. Likewise Capacity, an element of the first consumer principle component indicating the ability of the software to keep pace with demand, may well be a baseline expectation for consumers.

In spite of some differences in the two response sets, it seems reasonable to conclude that the first principle component for both producers and consumers corresponds to the same factor. This factor appears to be related to whether the software product satisfies consumer requirements and expectations.

There is only slightly less agreement between producer and consumer samples for the second and third principle components. Simplicity and Communication were components of both the second principle component for the consumer articles and of the third principle component for producer data. Both dimensions deal with ease of use: Simplicity is how complicated the software is and Communication is how easy software inputs and outputs are to assimilate into a larger system. Simplicity was a component of the third principle component for consumers as well. The

Efficiency vector, also a component of the second consumer principle component, had an orientation opposite that of the other dimensions that formed the second principle component. This may suggest a trade-off between Efficiency and the other components of ease of use. Serviceability was a component of this factor in the producer data and it seems reasonable to interpret this dimension as an element of ease of use. These results seem to indicate the existence of a valid factor related to ease of use, but a higher priority for this factor among consumers than among producers.

Capability, Flexibility, Accuracy and Descriptiveness were the components of the second producer principle component. The definitions for Capability and Flexibility both suggest a factor that is an expression of variation. Capability and Flexibility had moderately high frequency responses in both sets of data, but were not identified among significant principle components in the consumer response. Structuredness and Stability also had relatively high frequency responses in both producer and consumer data sets, but, conversely, were not elements of any of the significant producer principle components. Structuredness, the pattern of software organization, may be a consumer indicator of ease of use, given its inclusion in the second principle component for consumers. Thus it seems that the third principle component relates to a general sense of technical soundness, perhaps best captured by the Capability construct.

Stability, Simplicity, and Perceived Quality were components of the third consumer principle component. These results suggest a construct of "first impression quality", because they all deal with the perception of quality immediately after a consumer uses the product. Within the context of our sample (consumer reports), this appears to be a valid conclusion. It is unclear whether this dimension is strictly an artifact of our sample type, or whether it would hold up in general. Our intuition is that it would hold up in other contexts--consumers often make software purchase decisions based on experimentation with a demonstration disk or a version available on the retail floor, so first impression quality would be a significant determinant.

In conclusion, the principal components analysis suggests some possible meta-dimensions of software quality: fulfilling customer requirements, ease of use, and capability (for producers); and fulfilling customer requirements, ease of use, and first impression quality (for consumers). In looking at where the two lists differ, consumers appear to be interested in what the software can do for them now (first impression quality), whereas producers appear to be interested in what the software can do for them in the future (capability).

Dimensions from *Characteristics of Software Quality* and "The Strategic Nature of Process Quality" had moderately-high responses in both the consumer and producer sample sets. Many of these have operations definitions in the source reference and additional operational definitions would be relatively straight forward to develop. Missing from the list of significant dimensions for both consumers and producers are dimensions related to SERVQUAL and the service aspects of software products. There was not an equal density of constructs from the original sources of nominal construct definitions, which could account for the service dimensions receiving a relatively small response. In the authors' opinions, a more likely explanation is that these aspects of software operation are not well recognized.

Computer use, in all sorts of endeavors from analysis to entertainment, is changing from the exceptional to the pervasive. The knowledge and expectations are dynamic elements that will change with education and the social evolution of the role of software. Software systems that once provided certain consumers with a unique edge in the development, delivery, and support of their own products are now becoming necessary for mere survival. As a result, the mix of consumers contains a smaller proportion of enthusiasts and a greater relative frequency of reluctant conformers. The focus among many consumers is shifting from individual productivity enhancement to software that facilitates interaction among individuals. It is increasingly unlikely that an individual in this situation will be able to dictate hardware and operating software

requirements to all of their collaborators. As a result, considerations like Synchronization may become more significant to consumers in the future.

Another dynamic element relevant to software quality is the locus of specific functionality in layered software systems. Infrastructure has migrated between operating system, application, and the emerging category of "middleware". Modification of one element of the operating software infrastructure can be the stimulus for a failure of another software component. Improvements in the hardware environment can also compromise the value of specific software products. For example, a variety of older applications cannot drive laser printers and the authors have a collection of games that run too fast to play on current hardware platforms. This indicates that many, if not most requirements of a software product are unknown and unknowable by the producers. Therefore, a model for software quality that relies on conformance to explicitly known requirements is clearly insufficient.

The expanding capability of computer hardware also effects software quality attitudes. New devices for human interaction with the computer are being realized under the banner of virtual reality. Software development tools and productivity are rapidly evolving. These developments fuel availability and consumer expectations for software products and strongly support the idea that software product quality is an entropic function. Without continued effort to refine and improve software products, quality declines with the advance of time.

The manifest content analysis results support the presumption that current software quality research and practice among producers is concerned primarily with make-to-order software and the development process, not the product. The Conformance category cited most frequently in producer article is strongly associated with the development process. The latent implication in most references to the Conformance construct was formal compliance with the previous abstraction level of product requirements. Generalization of physical world phenomena is the an

important element of theory construction [Kuhn, 1970]. It seems evident from the observed responses that current software engineering theory was constructed on the generalization that software is made to order. While not a complete classification scheme, the make-to-stock versus make-to-order distinction appears to be valid based on the preceding analysis. The classification scheme should prove to be more longitudinally stable than those previously proposed for software products.

6 Conclusions

Manifest content analysis appears to be a useful technique for theory construction and validation at a relatively low cost. The data available, with the described limitations, was nonetheless useful in validating several assumptions regarding software quality. The initial analysis step successfully provided a foundation for more intrusive and prospective data collection. The quantitative and qualitative analysis also suggested several avenues for further research.

Guttman scale construction [Gorden, 1977] for the software quality constructs would be a valuable contribution. Discovery, calibration, and verification of ordinal scales for a subset of the constructs would facilitate the use of more powerful statistical modeling techniques. Scale generation is a recognized means of empirical theory validation. This effort would facilitate a move from conceptual to operational definitions for the theory constructs. This would be very beneficial to the research of quality in all fields, not just software.

As previously indicated, the investigation of other explanatory factors towards the construction of a longitudinally stable software classification taxonomy would be a valuable contribution to software engineering. Factorial experimentation is a logical tool to use to measure the predictive significance of candidate classification schemes. The stability of a proposed phylogeny could be assessed subjectively given what has been learned from the failures of other classification schemes. Time offers the only objective evaluation of the longitudinal stability of such schemes.

The results of this investigation suggest that there are several dynamic elements that effect quality attitudes. Powerful mathematical tools exist for modeling dynamic systems. Models featuring interactions, delays, and feedback often produce unique and useful predictions. The importance of novel predictions in growth and improvement of theory is supported by several factors [Chalmers, 1990].

Survey research could be used in the verification of observed construct priorities. A Delphi panel could also be convened for this purpose. Confirmation of the observed priorities would also indicate priorities for future software quality research. Another candidate for related future research involves exploration of predictive validity. A non-exhaustive set of predictions that are consistent with the model framework would be posed and empirical methods used to collect data and test statistical hypotheses to confirm predictions. Cross-validation is the ultimate test of factor selection [Lohlin, 1992] and it is hoped that future research will include repetition of this experiment.

ACKNOWLEDGEMENTS

This work has been partially supported by Honeywell Solid State Electronics Center. We wish to thank editor Prof. George Easton and all other anonymous reviewers for their helpful comments.

BIBLIOGRAPHY

- Agresti A., *Categorical Data Analysis*. New York: John Wiley & Sons 1990.
- Albrecht J. , "Measuring Application Development Productivity", *Proceedings of IBM Applications Development Symposium*. GUIDE International, October 1979.
- Babbie R. , *The Practice of Social Research*, 2nd edition. Belmont: Wadsworth, 1979.
- Basili V. R. and Weiss D. M. , "A Methodology for Collecting Valid Software Engineering Data", *IEEE Transactions on Software Engineering*. IEEE Computer Society Press, November 1984.
- Boehm W., Brown J. R., Kaspar H., Liplow M., Macleod G. J., and Merrit M. J. , *Characteristics of Software Quality*. Amsterdam: North-Holland, 1978.
- Boehm W. , "Verifying and Validating Software Requirements and Design Specifications", *IEEE Software*. New York: IEEE Computer Society Press, January 1984.
- Books in Print*, Volume 6. New York: R. R. Bowker, 1992.

- Brooks, F.P.: *The Mythical Man-Month*, Reading, Addison-Wesley, 1975.
- Chalmers A. , *Science and its Fabrication*. Buckingham: Open University Press, 1990.
- Cho C. K. , *Quality Programming: Developing and Testing Software using Statistical Quality Control*. New York: Wiley, 1987.
- Crosby B. , *Quality is Free*. New York: McGraw-Hill, 1979.
- Curtis B, Sheppard S. B., Milliman P., Borst M. A., and Love T. , "Measuring the Psychological Complexity of Software Maintenance Tasks with the Halstead and McCabe Metrics", *IEEE Transactions on Software Engineering* March 1979. New York: IEEE Computer Society Press, 1979.
- Department of Defense , *DOD-STD-2167 Defense System Software Development NAVMAT 09Y*. Washington DC: Department of Defense, 1985.
- Devlin K. , *Logic and Information*. Cambridge: University Press, 1991.
- Dijkstra E. W. , "Go To Statement Considered Harmful", *Communications of the ACM*, New York: Association for Computing Machinery, 1968.
- Dreger B. , *Function Point Analysis*. Englewood Cliffs: Prentice Hall, 1989.
- Fagan M. E. , "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems Journal*, Number 3. July 1979.
- Fisher J. and Light W. R., Jr. , "Definitions of Software Quality Management," *Software Quality Management*. New York: Petrocelli Books, 1979.
- Garvin A. , *Managing Quality*. New York: Free Press, 1988.
- Gibbs, W.W., "Software's Chronic Crisis," *Scientific American*. September 1994, p. 86-95.
- Gorden L. , *Unidimensional Scaling of Social Variables*. New York: Free Press, 1977.
- Guaspari, J. , *I Know It When I See It: A Modern Fable About Quality*. New York: American Management Association. 1985.
- Halstead H. , *Elements of Software Science*. New York: Elsevier North Holland, 1979.
- Hofri M. , *Probabilistic Analysis of Algorithms*. New York: Springer-Verlag, 1987.
- Holsti R. , "Content Analysis", *Handbook of Social Psychology*, 2nd Edition (Lindzey, editor). Reading, Mass: Addison-Wesley, 1968.
- Hopcroft E. and Ullman J. E. , *Introduction to Automata Theory, Languages and Computation*. Reading, Mass: Addison-Wesley, 1979.
- Humphrey S. and Sweet W. L. , *A Method for assessing the Software Engineering Capability of Contractors, Preliminary Version, Pittsburgh: Software Engineering Institute, September 1987*. CME/SEI-87-TR-23, ESD-TR-87-186, ADA187230, September 1987.
- ISO 8402, International Standard ISO/CD 8402-1, *Quality Concepts and Terminology, Part One: Generic Terms and Definitions*, International Organization for Standardization, December 1990.
- ISO 9126, International Standard ISO/IEC 9126, *Information Technology--Software Product Evaluation--Quality Characteristics and Guidelines for Their Use*, International Organization for Standardization, International Electrotechnical Committee, 1991.
- Jones T. C. , *Measuring Software*. Software Productivity Research, 1990.
- Kemerer C. F. , "An Empirical Validation of Software Cost Estimation Models," *Communications of the ACM*. New York: Association for Computing Machinery, 1987.
- Kuhn S. , *The Structure of Scientific Revolutions*, 2nd Edition. Chicago: University of Chicago Press, 1970.
- Loehlin J.C. , *Latent Variable Models*, 2nd Edition. Hillsdale: Lawrence Erlbaum Associates, 1992.

- McCall, J.A., Richards, P.K., and Walters, G.F., *Concepts and Definitions of Software Quality*, Springfield, 1977.
- McManus J.I: "The Heart of SQA Management: Negotiation, Compliance, Regression," *Handbook of Software Quality Assurance*. New York: Van Nostrand Reinhold, 1987
- Mistereck D. A., Anderson J., and Dooley K. , "The Strategic Nature of Process Quality," *Proceedings of Decision Science Institute*. Decision Science Institute, 1990.
- Morehead A. T. and Morehead P. D. , *The New American Roget's College Thesaurus in Dictionary Form*. New York: Signet, 1978.
- Pirsig M. , *Zen and the Art of Motorcycle Maintenance*. New York: William Morrow, 1974.
- Prell M. and Sheng A. P. , "Building Quality and Productivity into a Large Software System," *IEEE Software*. New York: IEEE Computer Society Press, July 1984.
- Ramamoorthy V., Prakash A., Tsai W. T., Usuda Y., "Software Engineering: Problems and Perspectives," *IEEE Computer*. New York: IEEE Computer Society Press, October 1984.
- Reifer J. , *State of the Art in Software Quality Management*. Reifer Consultants, 1985
- Schulmeyer G. , "Software Quality Assurance -- Coming to Terms," *Handbook of Software Quality Assurance*. New York: Van Nostrand Reinhold, 1987.
- Shen, S., "A Little Quality Time", *IEEE Software*. New York: IEEE Computer Society Press, September 1987.
- Software Engineering Technical Committee of the IEEE Computer Society, *IEEE Standard Glossary of Software Engineering Terminology, IEEE-STD-729-1983*. New York: IEEE Computer Society Press, 1983.
- Symons R. , *Software Sizing and Estimating: Mk II Function Point Analysis*. New York: John Wiley & Sons, 1991.
- Woolf B. (editor) , *Webster's New Collegiate Dictionary*. Springfield: G. & C. Merriam, 1977.
- Zeithaml A., Parasuraman A., Berry L. L. , *Delivering Quality Service*. New York: Free Press, 1990.

Access	ease of use of the program or its components (Bo)
Accuracy	correctness (Ba)
Aesthetics	how a product looks (G)
Capability	ability to keep variation within required limits (M)
Capacity	ability to produce at least at the rate of demand (M)
Cleanliness	cleanliness (M)
Communication	form of inputs and outputs is easy to assimilate into larger system (Bo)
Competence	possession of required skills and knowledge to perform service (Z)
Completeness	each part full developed (Bo)
Conciseness	no excess information is present (Bo)
Conformance	degree to which a products design and operating characteristics meet the stated requirements (G); "all parts present" portion of "Completeness" Characteristic (Bo)
Consistency	uniform notation, terminology, and symbology through each definition level (Bo)
Credibility	trustworthiness, believability (Z)
Descriptiveness	contains enough information to determine the objectives, assumptions, constraints, inputs, outputs, components, and status (Bo)
Durability	amount of use before replacement is more economical than repair (G)
Efficiency	rate of value and waste added per resource consumed (Bo, M)
Features	operating characteristics (G)
Flexibility	marginal cost to extend Features (M)
Independence	executable in hardware environment other than current one (Bo)
Legibility	function of program and components is easily discerned by reading the code (Bo)
Perceived Quality	impact of advertisements and references (G)
Performance	primary operating characteristics (G)
Precision	exactness of measure (Ba)
Responsiveness	willingness to help customers (Z)
Robustness	marginal cost of surviving unforeseen changes (M)
Safety	freedom from physical danger (M)
Security	freedom from risk or doubt (Z)
Serviceability	speed of repair (G)
Simplicity	how complicated (M)
Stability	predictability (M)
Structuredness	possesses a pattern of organization of its interdependent parts (Bo)
Synchronization	how well aligned different parts of the process are (M)
Tangibles	facilities, equipment, personnel, communication materials (Z)

Adapted from: (Ba) Babbie, 1979; (Bo) Boehm, 1978; (G) from Garvin, 1988; (Z) from Zeithaml et al, 1990; (M) from Misterek et al, 1990.

Table 1 Initial Software Quality Constructs

Quality Dimension	Producer	Consumer
	Accuracy, Capability, Completeness, Conformance, Flexibility, Serviceability, Stability, Structuredness	Capability, Communication, Completeness, Conformance, Features, Flexibility, Simplicity, Stability

Table 2 Most Frequent Quality Dimensions Found

Variable	PC1	PC2	PC3	
CONFORMA		.264	.076	.065
STRUCTUR	.157	-.059	.244	
CAPABILI		.139	-.251	-.168
STABILIT		-.235	-.140	.136
COMPLETE	.293	-.184	-.026	
ACCURACY		.254	-.268	.049
SERVICEA		.120	-.156	-.288
FLEXIBIL		.223	-.305	.111
DESCRIPT		.211	-.305	.137
SIMPLICI		.088	.156	-.352
EFFICIEN		-.243	-.235	-.135
COMMUNIC		-.032	.002	-.266
PERCEIVE		-.276	-.186	.040
CONSISTE		-.275	-.218	-.087
CAPACITY		-.271	-.160	.080
FEATURES		.103	.297	-.053
INDEPEND		.091	-.345	-.179
SECURITY		-.076	-.085	-.307
SAFETY		.205	.104	-.200
ROBUSTNE		-.177	.004	.297
CONCISEN		-.129	-.042	-.415
SYNCHRON		.282	-.192	.062
DURABILI		-.285	-.186	.077
PRECISIO		.029	.325	.045
LEGIBILI		.050	-.026	.319

Table 3 Producer Eigenvector Coefficients Theory Construct Reduction
Dimensions in relative frequency order

Variable	PC1	PC2	PC3
FEATURES	-0.287	0.069	0.031
CAPABILI	-0.178	-0.187	-0.049
FLEXIBIL	-0.154	0.086	0.148
CONFORMA	-0.271	0.092	0.123
COMMUNIC	-0.160	-0.368	0.080
SIMPLICI	0.100	-0.322	0.311
STABILIT	-0.145	0.237	0.289
COMPLETE	-0.286	-0.082	0.052
CAPACITY	-0.282	-0.078	-0.032
STRUCTUR	-0.141	-0.303	-0.249
INDEPEND	-0.227	0.014	0.206
EFFICIEN	-0.078	0.352	0.199
PERCEIVE	0.098	-0.015	0.362
ACCURACY	0.101	-0.322	0.118
DESCRIPT	-0.267	-0.163	0.047
SECURITY	-0.278	0.066	0.035
DURABILI	0.117	0.039	0.336
ROBUSTNE	-0.268	-0.150	0.004
SAFETY	-0.064	0.367	0.086
SERVICEA	-0.291	-0.024	0.005
SYNCHRON	-0.252	-0.049	-0.084
CONSISTE	0.125	-0.285	0.283
LEGIBILI	0.109	-0.201	0.048
AESTHETI	0.201	-0.028	0.257
CONCISEN	0.127	0.035	-0.341
PRECISIO	0.048	0.034	-0.290

Table 4 Consumer Eigenvector Coefficients
Dimensions in relative frequency order

APPENDIX A PUBLICATIONS SELECTED FOR MANIFEST CONTENT ANALYSIS

Producer Publications

1. Becker W: "EWSD software testing strategy ensures system quality", *Siemens Telcom Report*, Volume 11 Issue 6. Berlin: Siemens, Nov.-Dec. 1988.
2. Braun RL, Dunn RH, Erickson RL, Hon SE: "Software quality program requirements", *IEEE International Conference on Communications - BOSTONICC/89 - World Prosperity Through Communications*, Volume 1. New York: IEEE, 1989.
3. Chizhov S, Karpovskij E: "Software quality analysis", *EOQC Quality*, Volume 33, Issue 3. Rotterdam: EOQC, Sept. 1989.
4. Dixon A: "Software quality-a management strategy", *Software World*, Volume 21, Issue 2. Elmsford: Maxwell Scientific, 1990.
5. Hausen H L: "Yet another software quality and productivity modeling...-YAQUAPMO", *Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences*, Volume II. New York: IEEE Computer Society Press, 1989.
6. Ipoly T: "The practical use of quality management software measurement techniques," *First European Seminar on Software Quality Proceedings*. Paris: AFCIQ, 1988
7. Meredith DC: "Selling 'doing it right the first time' (software quality)," *System Development*, Volume 9, Issue 6. Phoenix: Applied Computer Research, June 1989.
8. Myers M, Kaposi AA: "Towards quality assurance of specifications by measurement", *Second European Conference on Software Quality Assurance*. Oslo: Norwegian Computer Society, 1990.
9. Nevalainen R: "Methods and tools for productivity and quality analysis in software projects", *VTT Symposium 104. Software Development Trends. Joint Finnish-Soviet Software Symposium*. Helsinki: Espoo, 1989.
10. Pendic Z, Kostic S, Grahovac J and Kovacevic L: "Incorporation of software logistics parameter-a way to software quality," *RELECTRONIC '88. 7th Symposium on Reliability in Electronics*. Budapest: OMKDK Technoinform, 1988.
11. Rannem S, Hung E: "On factors contributing to quality of nuclear control computer software", *Canadian Nuclear Society. 10th Annual Conference*, 1989. Proceedings, Volume 2. : Canadian Nucl. Soc, 1989.

12. Redig G, Swanson M: "Control Data Corporation's Government Systems Group Standard Software Quality Program", *Proceedings of the IEEE 1990 National Aerospace and Electronics Conference, NAECON 1990, Volume 2*. New York: IEEE, 1990.
13. Ronchi S, Martignano M, and Borghesi, M: "Expert systems software quality problems," *First European Seminar on Software Quality Proceedings*. Paris: AFCIQ, 1988.
14. Thompson JB: "The use of a quality assurance tool in the teaching of software engineering principles", *IEE Colloquium on 'The Teaching of Software, Engineering-Progress Reports'*. London: IEE, 1991.
15. Youll DP: "Achieving high quality software (automotive electronics)", *Proceedings of the Institution of Mechanical Engineers, Seventh International Conference, Automotive Electronics*. London: Mech. Eng. Publications, 1989.

Consumer Publications

- I Sheldon KM: "Lotus Improv for Windows," *Byte*, Volume 18, No. 2. Peterborough: McGraw-Hill, February 1993, pp. 52-53.
- II Udell J: "FoxPro 2.5 for DOS and Windows," *Byte*, Volume 18, No. 2., Peterborough: McGraw-Hill, February 1993, p 53.
- III Yager T: "The Second Premiere," *Byte*, Volume 18, No. 2., Peterborough: McGraw-Hill, February 1993, pp. 205-206.
- IV Yager T: "BASIC Breakthrough," *Byte*, Volume 18, No. 2., Peterborough: McGraw-Hill, February 1993, pp. 207-208.
- V Walsh B: "Correspondence That Looks Good Globally," *Byte*, Volume 18, No. 2., Peterborough: McGraw-Hill, February 1993, pp. 219-220.
- VI Berger I: "XTree 2.5 for MS-DOS and XTree for Windows," *Computer Buyer's Guide*, Volume 6, No. 2. New York: Harris Publications, Winter 1993, pp. 84-86.
- VII Finnie S: "PC Tools for Windows: The Ultimate Desktop," *PC Computing*. Boulder: Ziff-Davis Publishing, February 1993, pp. 38-41.
- VIII Taylor W: "MTV Meets the PC: Microsoft's Ambitious Video for Windows," *PC Computing*. Boulder: Ziff-Davis Publishing, February 1993, pp. 66,68.
- IX Rosch W: "PC/Computing Face-Off: Norton Utilities 6.01 vs. PC Tools 8.0," *PC Computing*. Boulder: Ziff-Davis Publishing, February 1993, pp. 314-319, 322-330.

- X Dragan R: "ObjectVision PRO: Quick, Robust Windows Prototypes," *PC Magazine*, Volume 12, No. 3. New York: Ziff-Davis Publishing, February 9, 1993, p 40.
- XI Dejean D: "askSam: Free-form Data Manager Gains GUI Strength," *PC Magazine*, Volume 12, No. 3. New York: Ziff-Davis Publishing, February 9, 1993, p 51.
- XII Karney J: "Systat for Windows: All-in-One Functionality," *PC Magazine*, Volume 12, No. 3. New York: Ziff-Davis Publishing, February 9, 1993, p 55.
- XIII Neuhaus T: "Ease the Pain of Graphics Programming with PCX Toolkit for Windows," *PC Magazine*, Volume 12, No. 3. New York: Ziff-Davis Publishing, February 9, 1993, p 75.
- XIV Simone L: "High End DTP: The Changing Of the Guard?" *PC Magazine*, Volume 12, No. 3. New York: Ziff-Davis Publishing, February 9, 1993, pp. 231-234, 242, 244, 246-249, 252, 256, 259-260, 262, 268-269.
- XV Stone MD: "Workgroup Scheduling: Have Your Computer Call My Computer," *PC Magazine*, Volume 12, No. 3. New York: Ziff-Davis Publishing, February 9, 1993, pp. 271-274, 277, 280-282, 287, 293-295.
- XVI Boling D: "RCLICK: A Pop-Up Menu Program for Windows," *PC Magazine*, Volume 12, No. 3. New York: Ziff-Davis Publishing, February 9, 1993, pp. 353-355, 358-359.
- XVII Porter S: "A New I-dea in Mechanical Design," *Computer Graphics World*, Volume 16, No. 5. Westford: PennWell Publishing, May 1993, p 15.
- XVIII Steuer S: "Painting Up a Storm: Painter 2.0 forms a vital part of the digital artist's toolbox," *Computer Graphics World*, Volume 16, No. 5. Westford: PennWell Publishing, May 1993, pp. 61-62.
- XIX Leland J: "Video Like the Big Boys Do It: CoSA attempts to capture the sophistication of high-end video systems -- with software on the Mac," *Computer Graphics World*, Volume 16, No. 5. Westford: PennWell Publishing, May 1993, pp. 62-63.
- XX Simerman T: "A New Side to Explore: TDI's high-end package offers something unusual Interactive rendering," *Computer Graphics World*, Volume 16, No. 5. Westford: PennWell Publishing, May 1993, pp. 63-64.
- XXI Felici J: "PagePlus 3.0," *Publish*, Volume 9, No. 11. San Francisco: Integrated Media, November 1994, pp. 31-32.
- XXII Burger J: "Adobe Premiere 4.0," *Publish*, Volume 9, No. 11. San Francisco: Integrated Media, November 1994, p 33.
- XXIII Doerner: "KPT Bryce 1.0," *Publish*, Volume 9, No. 11. San Francisco: Integrated Media, November 1994, p 34.

- XXIV Faye D: "Intellihance," *Publish*, Volume 9, No. 11. San Francisco: Integrated Media, November 1994, pp. 36-37.
- XXV Swart M: "Adobe Dimensions 2.0," *Publish*, Volume 9, No. 11. San Francisco: Integrated Media, November 1994, p 38.
- XXVI Frank M: "DataEdit 2.1," *DBMS*, Volume 7, No. 10. San Mateo: Miller Freeman, September 1994, pp. 28, 32.
- XXVII Indermaur K: "Scribe Professional 4.0," *DBMS*, Volume 7, No. 10. San Mateo: Miller Freeman, September 1994, pp. 32-34.
- XXVIII Schnapp M: "RaSQL/B for FoxPro," *DBMS*, Volume 7, No. 10. San Mateo: Miller Freeman, September 1994, pp. 34, 36, 40.
- XXIX : "MS-DOS Version 6.2," *Infoworld*, Volume , No. . : , January 24, 1994, pp. 60, 62, 66, 68.
- XXX : "PC-DOS Version 6.1," *Infoworld*, Volume , No. . : , January 24, 1994, pp. 60, 64, 66, 68.
- XXXI Marshall P: "Norton Desktop for Windows sets new standard," *Infoworld*, Volume , No. . : , January 24, 1994, pp. 66, 68.
- XXXII Williams C: "Improv for Windows 2.0: Viewing, printing, structuring options make it truly dynamic," *Infoworld*, Volume , No. . : , January 24, 1994, pp. 70-71.
- XXXIII Williams C: "Commander Prism 1.1 quickly manages large models but can be difficult to use," *Infoworld*, Volume , No. . : , January 24, 1994, pp. 73-74.

APPENDIX B INSTRUCTIONS FOR MANIFEST CONTENT ANALYSIS OF SOFTWARE PRODUCT REVIEWS IN POPULAR PERIODICALS

Reviewer Name: _____
Review Date: _____
Generator Seed: _____ Sample Number: _____
Publication Name: _____
Publisher: _____
Publication Date: _____
Product Review pages _____

General instructions: You are asked to analyze several publications. Try to identify a common setting for each analysis session. This includes location, lighting, time of day, and background noise. Try to minimize other distractions. Schedule occasional, short breaks to refresh yourself. Analysis of each publication should be completed in a single, uninterrupted session.

Product Review Analysis: The use of key words will be tallied. The tallying process covers all the text in all "software-only" product reviews in the periodical. Omit reviews less than one full page. All text should be analyzed including title, normal text, section headings, bibliographies, author biographies, and captions on figures and illustrations. Inset articles are omitted from analysis. The most recent issue of the selected periodical is to be used.

Key Word Tally: Familiarize yourself with the key word tally list. You will be identifying the key word and variations including prefixes, suffixes, tenses, and conjugations. If a key word appears to be used to describe a primary characteristic of the software or development methodology instead of its intended construct, do not tally that instance.

APPENDIX C MANIFEST CONTENT ANALYSIS CONSTRUCT CODING

accurate	ACCURACY	import	COMMUNICATION
adapt	ROBUSTNESS	impress	PERCEIVED QUALITY
advantage	PERCEIVED QUALITY	injury	SAFETY
aesthetic	AESTHETIC	integrate	COMPLETENESS
appeal	AESTHETIC	interim	STABILITY
approximate	PRECISION	legible	LEGIBILITY
architecture	INDEPENDENCE	maintain	SERVICEABILITY
aspect	FEATURES	metaphor	DESCRIPTIVENESS
attribute	FEATURES	modify	FLEXIBILITY
beauty	AESTHETIC	optimum	EFFICIENCY
benefit	PERCEIVED QUALITY	option	FEATURES
brief	CONCISENESS	organize	STRUCTUREDNESS
capable	CAPABILITY	perpetuity	DURABILITY
capacity	CAPACITY	plain	LEGIBILITY
challenge	SIMPLICITY	platform	INDEPENDENCE
classify	SECURITY	portable	COMMUNICATION
clear	LEGIBILITY	precise	PRECISION
cohesive	SYNCHRONIZATION	quirk	CONSISTENCY
communicate	COMMUNICATION	reasonable	ACCURACY
complete	COMPLETENESS	release	STABILITY
complex	SIMPLICITY	repair	SERVICEABILITY
compose	STRUCTUREDNESS	repeatable	CAPABILITY
concise	CONCISENESS	require	CONFORMANCE
confidence	SECURITY	resource	CAPACITY
configure	FLEXIBILITY	robust	ROBUSTNESS
conform	CONFORMANCE	round	PRECISION
congruous	CONSISTENCY	safe	SAFETY
consistent	CONSISTENCY	secure	SECURITY
constraint	CAPACITY	service	SERVICEABILITY
control	CAPABILITY	simple	SIMPLICITY
coordinate	SYNCHRONIZATION	specify	CONFORMANCE
correct	ACCURACY	stable	STABILITY
crowd	LEGIBILITY	standard	STABILITY
custom	FLEXIBILITY	structure	STRUCTUREDNESS
damage	SAFETY	succinct	CONCISENESS
danger	SAFETY	synchronous	SYNCHRONIZATION
dependent	INDEPENDENCE	terse	CONCISENESS
descriptive	DESCRIPTIVENESS	test	CONFORMANCE
detail	DESCRIPTIVENESS	uniform	CONSISTENCY
deviation	CAPABILITY	update	DURABILITY
device	INDEPENDENCE	upgrade	DURABILITY
diagnosis	SERVICEABILITY	valid	ACCURACY
diverse	ROBUSTNESS	value	PERCEIVED QUALITY
doubt	SECURITY	whole	COMPLETENESS
durable	DURABILITY		
efficient	EFFICIENCY		
elegance	AESTHETIC		
eliminate	EFFICIENCY		
embed	SYNCHRONIZATION		
enable	CAPACITY		
entire	COMPLETENESS		
exact	PRECISION		
explicit	DESCRIPTIVENESS		
export	COMMUNICATION		
feature	FEATURES		
flexible	FLEXIBILITY		
form	STRUCTUREDNESS		
frustrate	SIMPLICITY		
future	ROBUSTNESS		
idle	EFFICIENCY		

