NORTH-HOLLAND

Software Quality Assurance: An Analytical Survey and Research Prioritization

Arun Rai

Department of Decision Sciences, College of Business Administration, Georgia State University, Georgia, Atlanta

Haidong Song

Software Quality Consultant, San Francisco, California

Marvin Troutt

Department of Management, 214 Rehn Hall, Southern Illinois University at Carbondale, Carbondale, Illinois

We present an overview of the Software Quality Assurance (SQA) research domain. An extensive review of the literature was conducted to identify areas that are being currently investigated or have received attention from the research community. Articles appearing in outlets appropriate for software and information engineering were considered. Our categorization scheme includes four key dimensions: technical, managerial, organizational, and economic. These primary dimensions were deduced from the literature, and sub-dimensions were induced to lead to a finer categorization scheme. We present a summary of the content and methodological orientation of present research. In its present state, the SQA domain has largely drawn upon principles and theories from other reference areas but their integration with the actual task and technology context is still in a rudimentary stage. This is especially true for research dealing with organizational and managerial issues. Further, there is limited examination of related thematic issues across the technical, managerial, organizational and economic strands. We conclude by recommending directions for future research. © 1998 Elsevier Science Inc.

1. INTRODUCTION

Quality of products and services is greatly emphasized in today's society, and we have seen significant improvements in quality levels during the last few decades. The movement was triggered by Japanese firms putting a high priority on quality and using it as a means to gain competitive advantage in global markets. However, no systematic theories have evolved for the management of the quality of software and computer systems. "If a company adopted the level of quality of current software in manufacturing, it would go out of business tomorrow" (Cho, 1987). This statement, to a certain extent, portrays the status of software quality commonly developed by information departments in organizations. With an increasing recognition that computer systems are a competitive necessity for modern businesses (Crosby, 1980), it is not surprising that software quality is becoming an important topic for both practitioners and researchers.

Address correspondence to Dr. Arun Rai, Georgia State University, College of Business Administration, Department of Decision Sciences, Atlanta, GA 30303. E-mail: arunrai@gsu.edu

J. SYSTEMS SOFTWARE 1998; 40:67–83

655 Avenue of the Americas, New York, NY 10010

0164-1212/98/$19.00
PII S0164-1212(97)00146-5

Over the last decade, there has been a substantial rise in the research on and practice of software quality assurance. This paper provides an overview of the current research status and an analysis of the present state of knowledge in the area of software quality assurance. An extensive literature survey was conducted for this purpose. The articles identified were systematically classified into suitable categories. We first present the categorization scheme employed and the rationale for the scheme. We then go on to discuss the articles in each category, thrust of research to date within each identified category, and issues not currently addressed. This in turn forms the basis for our recommendations for future research.

2. ESTABLISHING THE DOMAIN

When discussing the research work in any domain, there are two important questions to consider:

1. What topics define the domain of the area of study?
2. What constitutes research within the domain? (Dickson and DeSanctis, 1990)

Henderson and Cooprider (1990) define production, coordination and organization as three dimensions for the development of computerized systems. The production dimension deals mainly with technical aspects; coordination focuses on managing the interrelationships and dependencies between activities and people; organizational technology deals with the environment for the technology and associated processes. In this paper we analyze the software quality assurance domain through a revised version of these dimensions—technical, managerial and organizational. Further, given the increasing importance of justifying investments in information technology (Dos Santos, 1991) and quality initiatives (Crosby, 1980), economics of software quality assurance has been included as a separate fourth dimension. The classification system employed in this study is schematically shown in Figure 1.

A review of the articles was used as a basis to identify sub-dimensions under each of the above primary dimensions. Software quality characteristics, quality assurance techniques, and software quality metrics were three sub-dimensions that emerged under the technical dimension. Characteristics of

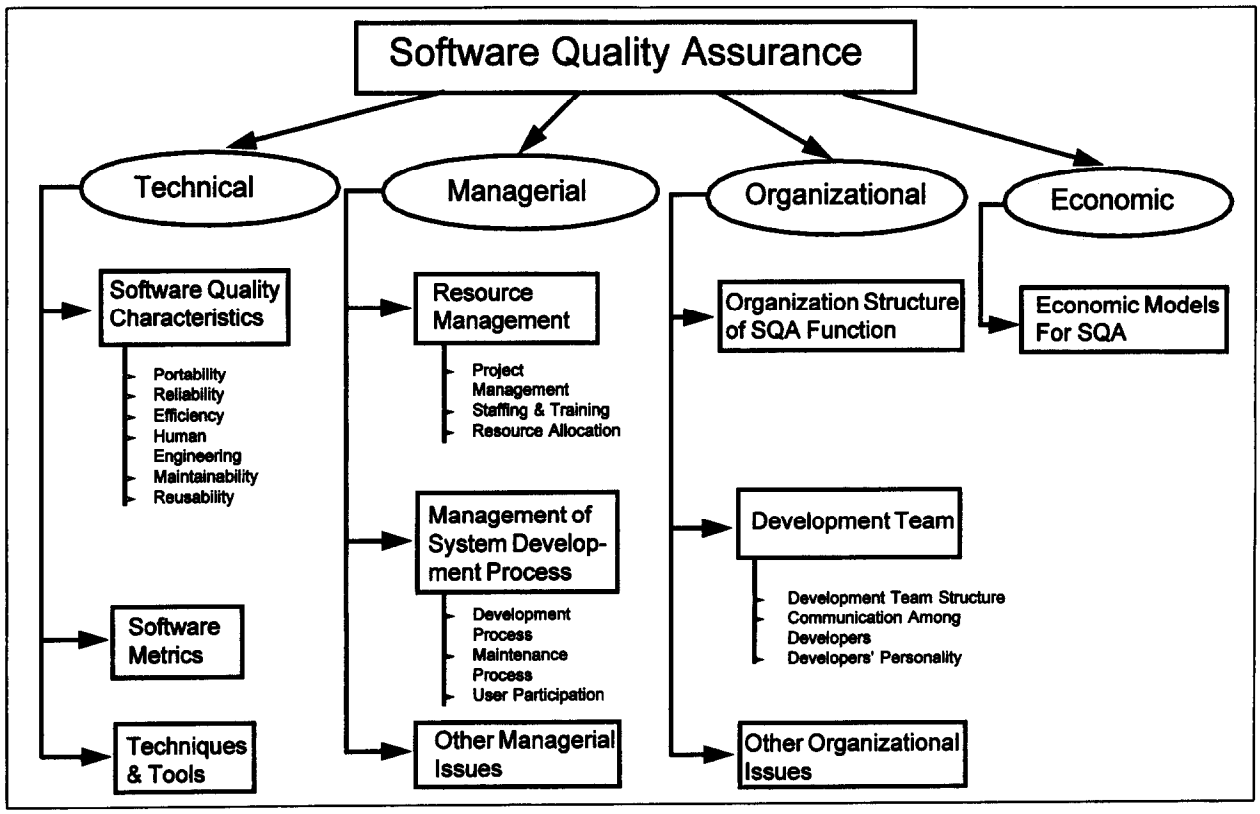

Figure 1. Classification scheme for software quality assurance.

software quality are adopted from Boehm, Brown, Kasper, Lipow, MacLeod and Merrit (1978) with some elaborations to accommodate for emergent themes such as reusability.

Under the managerial dimension, we identified two sub-dimensions: resource management, and management of the development process. The organizational dimension encompasses the organization of software quality assurance (SQA) activity, and the structural and behavioral characteristics of the development team. Two sub-dimensions are identified under the organizational dimension: organizational structure of SQA function, and characteristics of the development team. The fourth dimension—economics of software quality assurance—focuses on the cost and benefits associated with software quality assurance activities and models used to study them. Further elaboration of these sub-dimensions are included in the classification scheme to illustrate the emphasis of research efforts based on analysis of the literature.

3. STUDY METHODOLOGY

The articles included in this study were gathered using two methods. First, a key word search was employed using the *WILSONDISC* online index which includes over 1000 journals in various areas such as business, science, and medicine published during the past 10 years (Version 2.3.1, The H.W. Wilson Co.). Software Quality Assurance was used as the key word for the search.

Second, several leading journals in the areas of software quality, software engineering, and information systems were searched manually to identify articles relevant to our classification system. The journals were scanned for the 1980 to 1994 time period. A total of 401 articles were identified from these sources. The distribution of the articles by journals is shown in Table 1 and Figure 2 shows the distribution of articles by the primary topic areas.

In assessing disciplinary development of a field, it is useful to assess the methodological orientation of extant literature. Areas that are in their infancy are normally characterized by sporadic borrowing from other areas that can serve as appropriate reference disciplines. This stage of disciplinary development is typically characterized by a growth of frameworks and conceptual models. With time, researchers move to a "testing" mode and increasing amounts of empirical research are typically reported. Theories are refined to become more global and accommodate for appropriate contingencies. For example, fragmentation of results in the technological innovation area led researchers to focus substantial attention on resolving contradictory results.

Table 1. Journals Reviewed for SQA Research

Title of the Journal	Number of Articles
Journal of Systems and Software	67
Software Engineering Journal	33
IEEE Transactions on Software Engineering	28
IEEE Software	28
Software Quality Journal	48
Communications of the ACM	77
Management Science	6
MIS Quarterly	29
J. of Management Information Systems	17
Quality Progress	8
Information & Software Technology	25
Software: Practice & Experience	5
Other Journals	30

Each journal in the "Other Journals" category has at most three articles concerning software quality assurance.

Given the above discussion, it is useful to assess the methodological orientation of specific strands that characterize the SQA domain. Figure 3 presents the distribution of articles by methodological orientation and Figures 4 through 7 present the distribution of articles over different time periods indicating the trends in research activity for specific issues.

4. THE TECHNICAL DIMENSION OF SQA

4.1. Software Quality Characteristics (114 articles)

A second level of sub-dimensions was employed by drawing upon the Boehm et al. (1978) framework for the characteristics of software quality. In addition to the dimensions identified by Boehm et al., two other dimensions—efficiency and reusability—were included in this study to accommodate emergent trends. All the articles identified as dealing with software quality characteristics fit well into this elaborated framework.

4.1.1. Portability (3 articles). For portability, which means that software can be operated easily and well on different computer configurations (Boehm et al., 1978), only three articles were identified (Feinawer, 1991; Halasz, 1988; Wilden et al., 1991). Today, there are software packages which perform the same tasks on different computer configurations. With advances in data communication technology and the proliferation of concepts such as electronic data interchange (EDI), the focus has shifted to data exchange between different machines. A close look

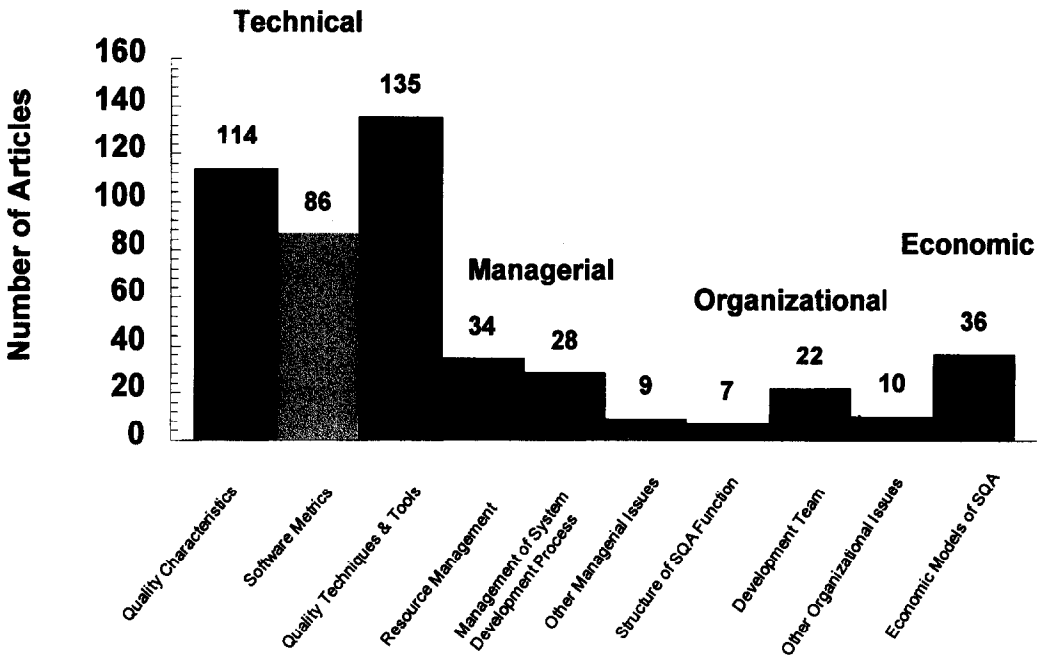

Figure 2. SQA literature topical coverage.

at the limited research indicates that there is also a call to shift emphasis from assessing portability of a program on different host machines, operating systems, compilers and compiler options (Feinawer, 1991) to the study of specification-level portability between different application programs (Wilden et al., 1991). Therefore, although the ability to process data from and transfer data to other computer configurations is important, it is possible that researchers recognize portability as an ongoing de facto vendor initiative and as a result have not actively studied this subject.

4.1.2. Efficiency (2 articles). Efficiency is a major concern of software developers and an important topic associated with system design and develop-

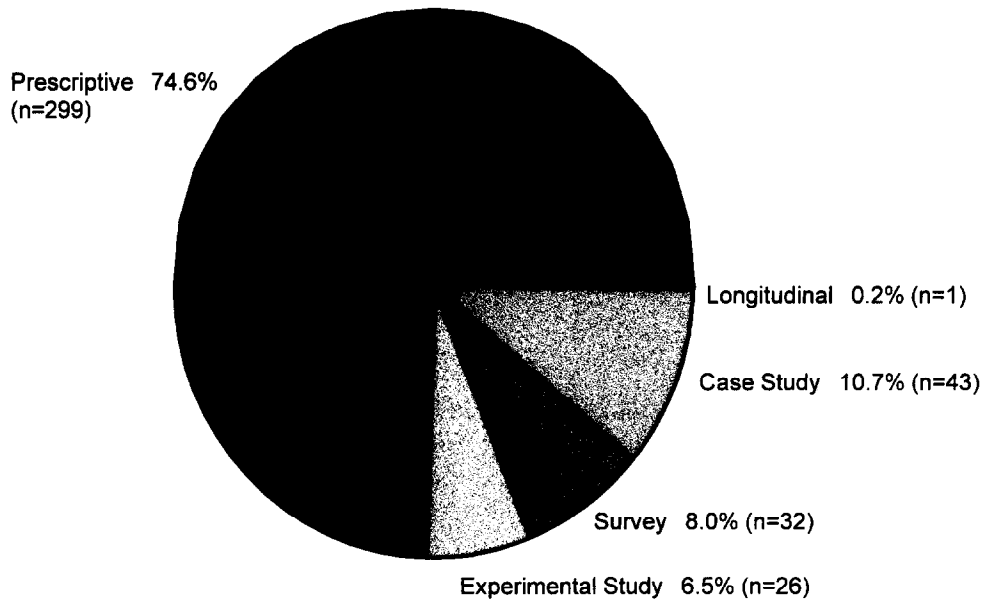

Figure 3. Research methodologies used.

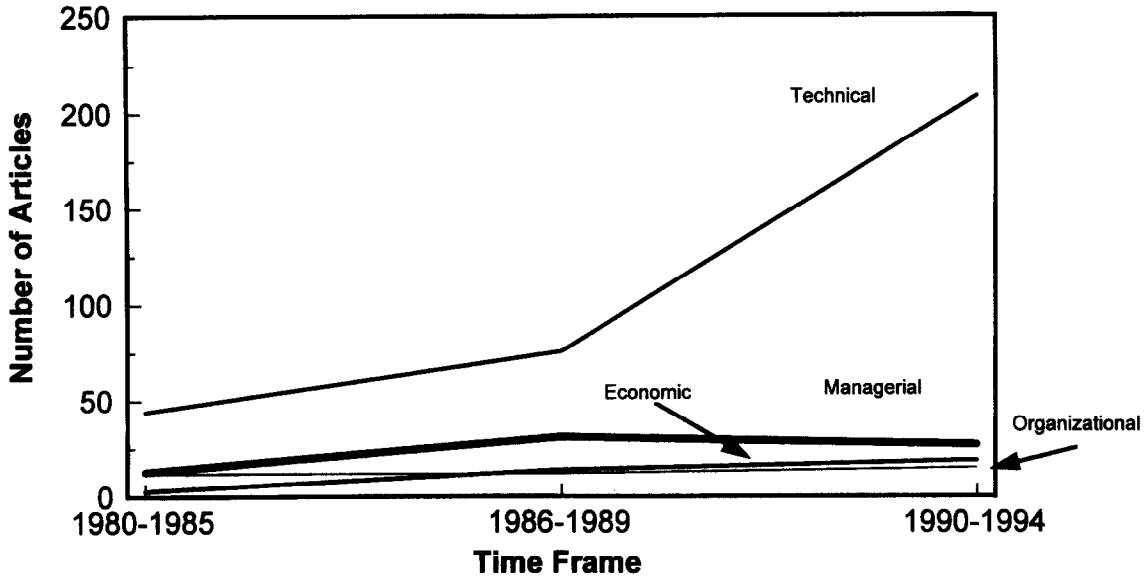

Figure 4. Evolution of research on SQA.

ment. However, the issue has been largely overlooked by researchers examining software quality. One possible reason is that hardware performance has exponentially increased in the last several years. This improved performance when coupled with decreases in the unit cost for computing significantly diminish the hurdles to processing operations in a timely manner. Therefore, the needs and attention for research on efficiency in the conventional sense has been somewhat diminished. We point out that the ease with which data can be accessed is treated under the human engineering sub-dimension. Clearly, difficulties in accessing data can negatively influence the efficiency of the computer system.

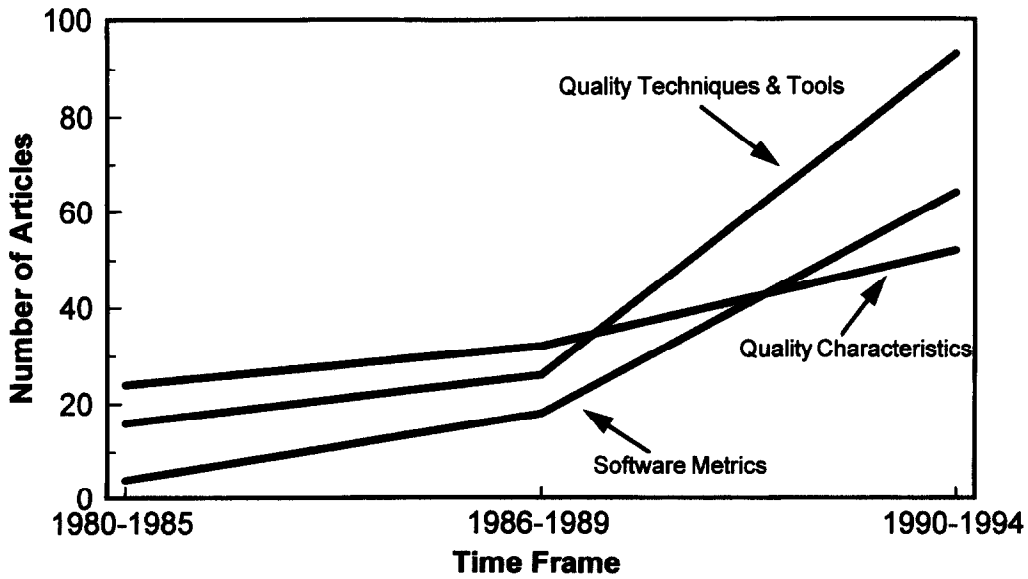

Figure 5. Evolution of research within technical dimension.

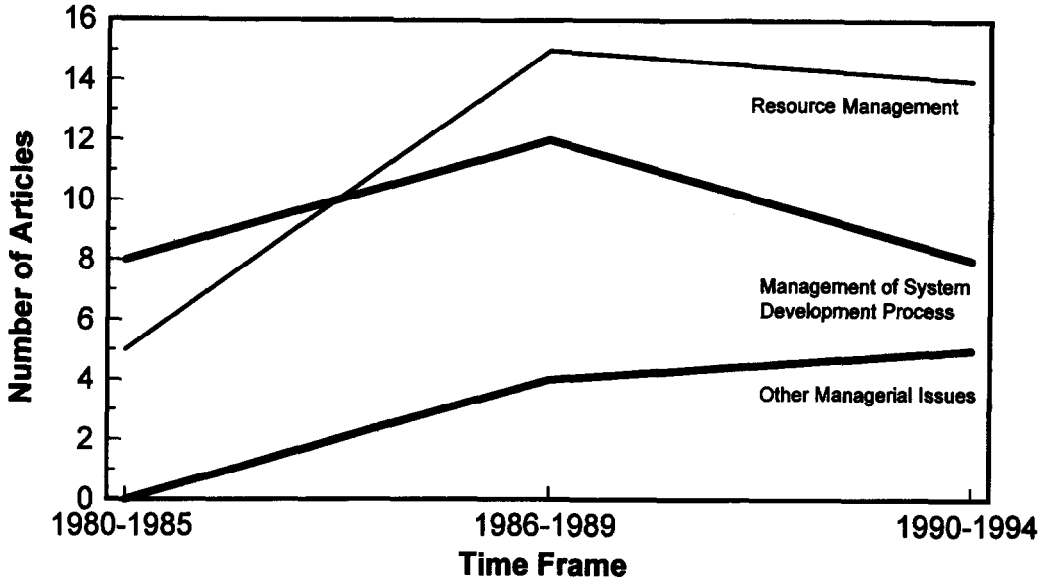

Figure 6. Evolution of research within managerial dimension.

4.1.3. Human Engineering (29 articles). Human engineering describes how easily a system can be understood and used. Among all the software quality characteristics, this is the only one that can be considered as user oriented. This issue has been consistently regarded as very important for the acceptance and the success of computer systems. The number of articles identified in this category reflects the importance of the issue at hand. Among the works on human engineering, a lot of research has focused on user cognition (Lucas and Nielson, 1980), behavioral (Napier et al., 1989), attitudinal (Carrol and McKendrea, 1987), and anthropometric characteristics. The research methodologies vary and include direct contact with intended or actual users, interviews, surveys, and experiments (Lucas and

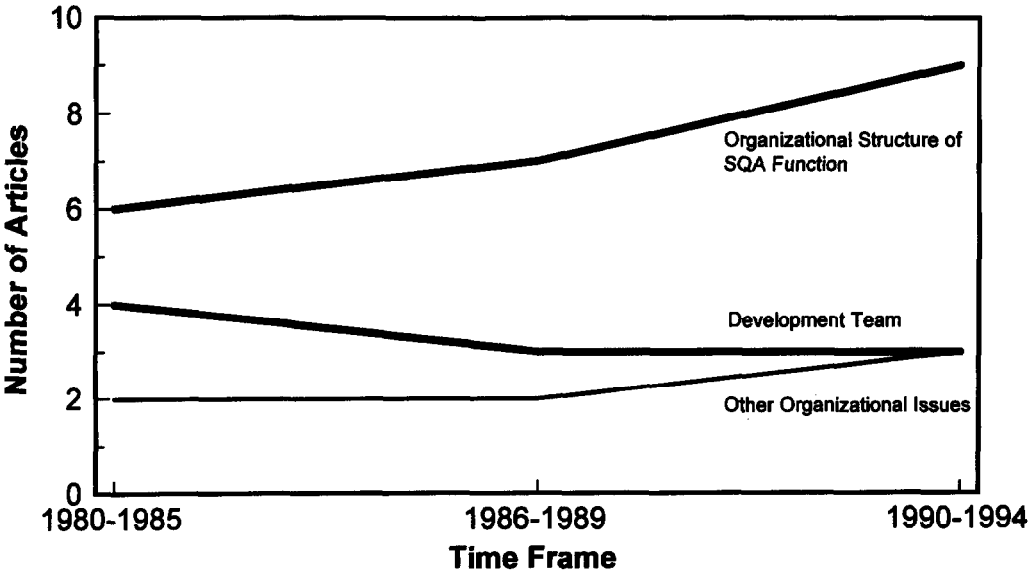

Figure 7. Evolution of research within organizational dimension.

Nielson, 1980; Napier et al., 1989). From these studies, some relationships between user characteristics and interface design have been developed. However, the measurement of the human engineering aspects has been primarily based on users' reaction to the software. Objective measurements of user behavior in a specific interaction of task and software technology have not been defined or used.

4.1.4. Reliability (38 articles). Reliability is the most extensively studied software quality characteristic. This is understandable given the importance of the issue in the case of mission-critical systems such as nuclear plant control systems (Feinawer, 1991), telephone switching systems (Rush et al., 1990) and others (Allot, 1992; Hansen, 1990). Consequently, reliability is suggested as one measure of computer systems success (Zahedi, 1987).

There has been a noticeable growth in research on the subject since 1986. This is understandable as business and society become increasingly dependent on computer systems. The failed CONFIRM project of American Airlines and the failure of the routing systems of AT & T on the east coast turned out to be management nightmares for the firms concerned. By one estimate, financial institutions have between 20% and 80% of their cash on-line at a given point in time.

Most articles either discuss methods and approaches to achieve software reliability or propose models for the measurement, analysis, and prediction of software reliability. It is suggested that a software engineering approach should be applied to every phase of the development life cycle in order to predict, measure and manage reliability (Musa and Everett, 1990). Various techniques such as Markov processes (Siegrist, 1988) and software fault trees (Leveson et al., 1991) have been suggested as modeling approaches to study software reliability. A compilation of commonly used modeling methods and their applicability is provided by (Ramamoorthy and Bastani, 1982).

4.1.5. Maintainability: Understandability, Testability, and Modifiability (20 articles). With maintenance cost accounting for more than 60% of the overall cost in an application system's life cycle, the need for attention on software maintainability is exigent. In the Boehm et al. framework (Boehm et al., 1978), maintainability is divided into three subdimensions: testability, understandability and modifiability. In all but one article, the maintenance of software has been treated as a singular concept. Only the understandability sub-dimension is explicitly discussed (Tenny, 1988). The other two subdimensions are not explicitly discussed in any of the articles, reflecting inadequate attention by software quality researchers to the multidimensional and complex nature of the "maintainable software" construct.

In general, the content of published work in this area is quite dispersed. The topics examined include software maintenance and its relation to personnel and procedures (Edwards, 1984); the impact of maintenance practices on software quality (Collofello and Buck, 1987); the perceptions of EDP professionals on factors that impact software maintainability (Kim and Westin, 1988); and the effect of software attributes (Rombach, 1987) and programmer's skills and programming styles (Vessey and Weber, 1983) on software maintainability. Behavioral issues such as the use of a communication-oriented approach have been suggested (Cashman and Holt, 1980) as a means to structure the software maintenance environment.

4.1.6. Reusability (17 articles). The concept of reusability has been explored by some firms and a few case studies are reported in the recent literature (Aptc et al., 1990; Banker and Kauffman, 1991). Software reusability is regarded by some researchers as the key to improved software development, productivity and quality (Biggerstaff and Richter, 1987). Some Japanese software producers have applied the concept of reusability and report significant improvements in both productivity and quality (Cusumano, 1989). However, there is very limited work on evaluating the effectiveness of alternative reusability implementation strategies. One exception is a set of economic models of reuse (Gaffrey and Durek, 1989) that relate system development productivity to the proportion of reuse and the cost of developing reusable components. In addition, Pfleeger and Bollinger (1994) presented a set of techniques to be used for modeling and assessing costs associated with software reuse. Methodologies and techniques to achieve reusability are also illustrated in other articles (Biggerstaff and Richter, 1987; Karimi, 1990; Prieto-Diaz and Freeman, 1987).

4.2. Software Metrics (86 articles)

Software metrics are critical to any quality initiative in the area and are used as a mechanism to quantify and measure some aspects of software quality (Ince, 1990). Keeping in line with the "management by fact" principle associated with Total Quality

Management (TQM) systems, it is important that appropriate metrics be designed and used. These measures should reflect both software product and development process quality. The increasing importance of software metrics is shown by the increase in published research articles over time. During the period from 1980 to 1985, there were only four articles dealing with software metrics. However, the figure rose to 17 in the period from 1986 to 1989 and to 64 for the 1990 to 1994 period.

Most of the articles classified into this category were found to be descriptive in nature. More specifically, the topics addressed include a general discussion on metrics (Grumen, 1991; Ince, 1990), the application of metrics to software quality assurance (Carpenter and Murine, 1984; Murine, 1988; Nenz, 1985; Siegel, 1992), metrics appropriate for different phases of the software development life cycle (Farbey, 1990; Heitkoetter et al., 1990; Kitchenham and Linkman, 1990; McCabe and Butler, 1989; Siegel, 1992), and productivity metrics (Yu et al., 1990).

Some authors have suggested that a standard set of metrics are needed for software quality measurement (Buckley and Poston, 1984). Others have pointed out that such a goal is not attainable. Recently, Poore (1988) proposed a theory to derive localized software quality metrics, and the same has been tested through two field experiments (Binder and Poore, 1990; Trammel and Poore, 1992). The results of these experiments support Poore's original thesis that IS managers should design localized software quality metrics rather than search for a "commonly accepted metrics set." Since software metrics are measures of software quality, they need to be validated prior to use. A comprehensive metrics validation methodology which includes six criteria has been proposed by Schneidwind (1992).

As with other products, the quality of the software product is tightly related to the development process. Considering only product quality is not enough as the process must also be monitored and controlled (Basili and Rombach, 1987). More than 60% of the articles have focused on product-oriented software metrics. There has been some attention by more recent works on process-oriented metrics (Bhide, 1990; Farbey, 1990; Harrison, 1988; Pfleeger and McGown, 1990; Reynolds, 1987). Some others have discussed both product- and process-oriented metrics (Hallonan et al., 1978; Kitchenham and Linkman, 1990). Moreover, the focal level of the metrics discussed to date tends to be solely at the unit level: for product metrics, the focus is on single module or single quality characteristics; for process metrics, the focus is on one project or one phase in the system development life cycle.

In contrast, there is growing emphasis in organizations to construct an enabling global information infrastructure. Construction of such an infrastructure has been recognized as among the top priorities of CIOs today (Niederman et al., 1991). A recent paper (Swanson et al., 1991) reported the construction of an Application Software Factory. Similar efforts by Japanese corporations have been reported too (Cusumano, 1989; Cusumano and Kemerer, 1990). There appears to be a clear gap in the present orientation of unit metrics and the increased emphasis on global information infrastructures.

4.3. Software Quality Techniques and Tools (135 articles)

The articles in this category deal with the techniques and tools that can be employed to control or improve software quality. Some articles in this category discuss the application of statistical techniques to software quality assurance (Ahituv and Zelek, 1987; Camuffo et al., 1990; Munson and Khoshgoftaar, 1992; Okumoto, 1985). Other articles cover a variety of techniques and methods to control and improve software quality using qualitative suggestions advocated by quality gurus such as Edward Deming (Miller, 1989; Zulter, 1988).

Some of the techniques and tools are adapted from existing approaches in general quality management such as statistical quality control methods, quality function deployment (QFD) (Van Treeck and Thackeray, 1991), and the application of a manufacturing process in systems development (Cusumano, 1989; Levendel, 1991). These adapted techniques and tools account for approximately 30% of the articles in this category. Other research efforts deal with techniques and tools that are unique to the system and software development process. Whether the techniques and tools are adapted from existing general quality management approaches or are unique to the system and software development process, most researchers focus on a specific quality characteristic or a particular phase of the development process.

5. THE MANAGERIAL DIMENSION OF SQA

Although 68 articles fall into this category, most of the articles are descriptive or prescriptive in nature. Among various topics in the resources management category, project management (23 articles), and development team staffing and training (10 articles) have received some attention. Boehm and Ross

(1989) developed a software project management theory called "Theory-W" where the primary task of the software project manager is to make winners of all parties involved in the development process. Swanson et al. (1991) demonstrate that there is a paradigm shift in project management when a transition is made to a software factory development approach. Other articles discuss project management as an approach to ensure productivity of personnel and quality of software products. Only one article (Carpenter and Hallman, 1985) provides a detailed discussion of the training content and process. Other articles discuss the need for training and their impact on quality in very general terms.

The research on management of the system delivery process includes topics such as management of the development process, management of the maintenance process and user participation. The general focus is on the application of quality assurance approaches to the development process, and on monitoring and controlling the development process. The application of general quality control/ assurance approaches used in other industries and various quality standards such as ISO9001 are suggested by several researchers as useful means to manage the system delivery process. For example, Kane (1992) described how TQM methodologies can be applied to software development and Rahman (1987) discussed the application of the quality circle concept in the context of the development process.

Some studies have focused on the impact of development methodologies on system quality (Alavi, 1984; Apte et al., 1990; Cerveny et al., 1986; Swanson et al., 1991). A few articles deal with quality assurance issues within individual phases in the development process. The use of CASE and other automated tools can help with the design phase, but requirement specification still remains a communication issue between the developers and users, and within the development team. Mantei and Teorey (1988) suggested the incorporation of behavioral techniques in the system development life cycle to enable effective management of the development process. In addition, user participation and involvement have been found to have a significant impact on management and quality of system delivery process (Dagwell and Weber, 1983; Franz, 1985; Gould and Lewis, 1985; Tait and Vessey, 1988).

6. THE ORGANIZATIONAL DIMENSION OF SQA

Under the organizational dimension of SQA we identified 39 relevant articles. Of these, seven deal with the structure of the software quality assurance function and 22 others deal with the characteristics of development teams including team structure, communication among developers, and developer's personality.

The articles relating to organizational structure of SQA have two distinct characteristics. First, they all deal with the placement of the SQA function or team in the organizational structure. Second, only two out of the seven articles were empirical in nature. The primary research question addressed by these articles is the independence of SQA teams from or affiliation with software development teams (Brelsford, 1988; Buckley and Poston, 1984; Grumen, 1991; Nenz, 1985).

The other 10 articles are very broad and cover a variety of organizational and behavioral issues, such as the level of motivation of developers to achieve high quality (Apte et al., 1990; Karimi, 1990; Kishida et al., 1987); the impact of the development environment on software maintenance performance (Bendifallah and Scacchi, 1987; Kim and Westin, 1988); a social dynamics perspective on user-analyst relationships (Newman and Robey, 1992); the centralization or decentralization of the control structure associated with IS planning and design (Henderson and Lee, 1992); communication in the development team (Poston and Bruen, 1987; Swanson et al., 1991); and the importance of organizational culture in achieving software quality (Kane, 1992).

7. THE ECONOMICS DIMENSION OF SQA

"Quality is Free" is an often quoted motto (Crosby, 1980). Yet the question of "What is it going to cost me?" is asked by many project managers whenever a quality goal is established. This is further confounded as the costs and benefits related to software and their impacts are more difficult to estimate than with other products and processes (Boehm, 1981).

In our assessment of the literature, we found 36 articles related to the economic dimension of SQA. Most of these articles provide general discussions of costs and benefits of software quality activities. Cost effectiveness of various software quality assurance practices are dealt with by Abdel-Hamid (1988), Barnes and Bollinger (1991), Levendel (1990), and Murine (1988). Hollocken (1986), Paughtrey (1988), and Zulter (1988) present a more detailed list of cost items related to software quality assurance, while Rivard and Kaiser (1989) looked at quality benefits descriptively.

A significant amount of effort has been devoted to establish a model for the cost and benefit involved in

the software development process (Grady, 1987; Grumen, 1991; Hollocken, 1986; Mukhopadhyay et al., 1992). However, these models are far from accurate (Kuster et al., 1990) and, therefore, only limited confidence should be placed in such estimates. In addition, only a handful of papers focus on the cost and benefit associated with software quality assurance activities. Others focus on the general cost and benefit of the overall development effort. However, estimation and analysis of costs and benefits of quality remains a significant issue. Only two articles have specifically looked at economic models and employed an empirical approach for the validation of these cost estimation models (Kemerer, 1987; Mukhopadhyay et al., 1992).

8. PRESENT STATUS AND FUTURE DIRECTIONS

8.1. Summary of Issues Examined

SQA research activity is dominated by descriptive and prescriptive writings (299 articles), with fewer empirical studies (102 articles). Furthermore, most reported empirical articles are exploratory in nature. This suggests that the SQA domain is still in an evolutionary stage with limited empirical and confirmatory research.

8.1.1. Technical Dimension. Only one article was found that dealt with efficiency from a quality perspective. Among all the other software quality characteristics, human engineering can be expected to grow significantly in importance, given the movement toward large-scale information infrastructures and the rapidly growing user base of computer systems. The interface design of interactive systems raises significant issues from a human information processing standpoint. Issues such as whether certain interfaces such as those in typical executive information system applications enforce cognitive biases have been recently raised (Rai et al., 1994).

Another interesting finding from the literature is that significant attention has been paid to the measurement, analysis, and prediction of software reliability (54% of the articles in this dimension). The majority of these articles view the reliability issue from the system developer's point of view. As reliability assessment should reflect the degree to which the software product performs intended functions correctly and satisfactorily (Boehm et al., 1978), an accurate assessment of reliability should involve user inputs.

Unlike other published work under the technical dimensions of software quality, about half the articles examining maintainability are empirical in nature. However, none of the empirical articles are longitudinal. Taken collectively, these articles examine the relationships between individual characteristics, software development characteristics and other technical attributes and software maintainability. We did not identify any study which examined these issues collectively or studied the interaction between these issues.

Implementing a development infrastructure designed around the principles of reusability calls for a fundamental rethinking of how software is developed. It can, in a sense, be construed as a prime example of business process redesign in software development, considering the fact that the ultimate reuse will include domain knowledge and development methodologies in addition to the software programs. Clearly, this is an area where significant work is needed to explore and understand the technical infrastructure, characteristics of process technology, organizational standards, and management practices required to move an IS organization toward a reusability-oriented environment.

There has been a steady increase in research efforts on software metrics and their use in the software development process for industrial applications (Andersen, 1992). However, the SQA literature does not provide frameworks to guide the selection and use of metrics. This area warrants future work as the central notion of quality can be argued to be "management by fact." Furthermore, the literature shows a paucity of research dealing with the deployment process of software metrics: How do these metrics originate in organizations? How are they used to monitor the development process and its management? The few articles in this area are either descriptive or based on single case studies, such as the experiences of Contel and Motorola with software measurement programs reported in (Daskalantonakis, 1992; Pfleeger, 1993). In order to fully utilize the benefits of software metrics, the dynamics of the deployment process and its relationship with contextual variables need to be explored.

In addition, metrics need to be developed to measure quality issues at levels higher than individual projects. Also, as IS organizations reconceptualize their systems delivery methods, the metrics used during transitory periods can be quite important. Consider an organization that is in the process of implementing reusability. If management continues to assess productivity based on lines of code produced or function-point analysis, they may get incorrect signals and misallocate resources, thereby negatively impacting otherwise well-conceived transition

plans. The metrics designed to promote reusability should recognize effort expended on parameterization of design and code.

Despite the research on software quality techniques and tools, software development does not employ quality control techniques as extensively as other production processes (Cho, 1987). A framework to guide the selection and application of techniques and tools in accordance with quality characteristics and development phases has not emerged. This situation is changing gradually with the emergence of some software factories in the U.S.A. and Japan (Cusumano, 1989). Little applied theory exists to systematically guide adoption of statistical techniques, methods, and automated tools, and assess the impact of different development approaches on product and process quality.

8.1.2. Managerial Dimension. The distribution of articles over time suggests that project management and management of the development processes have received relatively more attention from researchers. An assessment of the distribution in other areas does not reveal any significant trends. In general, the research focus has been on managing a single project or a specific development approach. There is no reported research on quality issues associated with infrastructure-oriented management practices (as opposed to project-oriented management practices). Furthermore, we did not identify any comparative studies of management approaches for different process technologies.

The research, in general, appears to have adopted managerial concepts from the TQM and organizational literatures. Management can play a major role in the "conversion effectiveness" of investments in such technologies to actual performance improvements. A similar theme has been expressed by Weill (1992) in his recent study examining the relationship between IT investments and organizational performance.

8.1.3. Organizational Dimension. The paucity of work in a single topic shows the diversity of interest on organizational issues among the researchers. The current knowledge can be characterized as fragmented with a need for consolidation and theory-building. The distribution of articles over time shows that there has been no growth in research activity in these areas. Moving beyond generic principles derived from TQM and organization theories will require that researchers address systematic differences in software process technology. The organizational dimension should be considered as an integral part of the effective adaptation and application of a specific process technology. As per Cooprider and Henderson (1990), organizational technology should be considered in conjunction with the production and coordination technologies used for software development. General organizational guidelines, while useful, have limited power in a prescriptive sense.

The managerial and organizational sub-dimensions reveal some common attributes. Both are broad and diffused and integrated theories do not exist presently. Also, there is a strong need for consolidation of existing fragments and theory building coupled with appropriate longitudinal empirical studies.

8.1.4. Economic Dimension. The economics of software quality assurance has received broad treatment primarily from a descriptive perspective. Few mathematical models have been developed. A better understanding of the costs and benefits of SQA and improvements to existing quantitative models should be useful to decision-makers. A look at the distribution of articles over time illustrates that there has been a significant increase in research efforts in the area during the recent two year period. It is plausible that IS managers are under increasing pressure to justify quality assurance programs/initiatives prior to experimenting with or embracing them. This notion is consistent with the ever-increasing pressure on senior IS executives to justify investments in information technology or manage information technology from a performance standpoint (Weill, 1992).

8.2. The Need to Build Thematic Bridges

There were some cases where the classification was problematic because themes explored crossed over. It is useful to provide a sense of such thematic bridges between the dimensions identified, as SQA calls for a managerial and organizational approach to the study of tools, techniques and technology.

There are a few thematic bridges between the managerial/organizational and the technical area. However, these bridges have to be substantially developed to come up with appropriate contingency theories. For example, while the placement/structure of the SQA function in general has received some attention, it is important to understand how the structure and role of the SQA function should differ for different development methodologies. As another example, the relationships between specific software characteristics and tools and techniques have not been explored in detail. Further, advances in one particular sub-dimension have not been factored in adequately in other sub-dimensions. The

recent advances in object-oriented systems have implications for both the software metrics and software characteristics sub-dimensions. Given the radically different approaches to building software and structure of software products under this approach, it is clearly important for researchers to carefully examine whether additional characteristics may be needed to define object-oriented software products and the relevance and adequacy of present metric systems from both a product and process standpoint.

Moreover, recent research shows that the conversion effectiveness of the managerial and organizational context can better explain the relationship between IT investments and organizational performance (Weill, 1992). Further, a recent study suggests and empirically demonstrates that the technological context of the IT investment should be addressed while studying the relationship with performance (Dos Santos et al., 1993). The evidence from the IT investment literature suggests that researchers in the SQA domain will benefit by considering such thematic bridges between the dimensions while studying performance impacts of SQA activities and techniques.

8.3. Integrating SQA with the Development Process

Consistent with the idea of building thematic bridges, we suggest that SQA should become an integral part of the development process. This approach recognizes quality as an ongoing effort during every step of the development process. Some work has been reported in this direction, noticeably the Capability Maturity Model proposed by Humphrey (1989) and his colleagues at the Software Engineering Institute. Software quality assurance is discussed with one or more development phases. However, the focus is on the assessment of the development process, and the overall linkage between software quality assurance and the development process has not been well explored. Among all the articles, we found 80 of them dealing with SQA and the development process in tandem. We note three themes directed at integrating SQA and the development process.

One theme linking SQA to the development process focuses on the application of general quality control/assurance approaches used in other industries. The techniques discussed include various quality standards such as ISO9001, and quality assurance activities such as quality function deployment and process control. However, the "fit" between these standards and techniques and particular software development processes has not been explored. For example, Hunter (1992) argued that many standards fail to take into account the essential differences, and occasionally the similarities, between software products and processes and other engineering products and processes. The applicability of such standards and techniques in software development still remains an unanswered question. Moreover, in light of rapid advances in software technologies, how standards and general quality assurance techniques will adapt to the change, and how the new technology will affect the quality practices in the system development process are areas for future research.

The second theme identified here focuses on "localized" quality practices. Specifically, quality assurance activities within individual phases in the development process, and techniques that can be used during the development process to improve software quality are examined. Many researchers and practitioners have argued that quality should be designed, not tested, into the software product. However, testing and various other review and control mechanisms are still the dominant approaches reported and discussed for ensuring quality. The coupling of testing to early stage activities, the usage of measurement, reusable components, etc., do not solve the problem entirely. What is still missing is the techniques and mechanisms to ensure quality from the very start and through the entire life cycle of the project. This calls for a revamping of the requirement specification and design process so that quality is embedded into the system from the beginning.

The third theme identified is an assessment of the impact of development methodology on quality. The discussion is mostly from a technical perspective, and managerial and organizational issues are not considered in tandem. However, the managerial and organizational aspects are likely to moderate the relationship between characteristics of the development methodology and the quality of outcomes. For example, as systems are built differently under alternate development approaches, it is reasonable to expect that the role of the SQA function should differ for these different process technologies. For instance, the structure and role of the SQA function would not be the same for systems built using the data-driven information engineering or structured systems analysis and design approaches.

Furthermore, the roles of programmer/analysts, content of communication between developers and users, and media deployed tend to be different un-

der different process technologies. For example, under the object-oriented approach, communication would focus on the definition of and relationships between objects. Under the data-driven information engineering approach, communication would mostly center on the data architecture and interrelationships between data entities. The nature of the process technology used for the construction of software significantly influences what can be referred to as the four Ws (why, what, who, where) and the H (how) of software production: *why* are certain steps needed in the process; *what* is produced during the intermediate steps; *who* is responsible for specific steps and outputs in the process; *where* will the activities be carried out and where intermediate outputs be stored; and *how* are all of the steps interrelated in an architectural sense so as to result in the final software product.

8.4. Implications of Shifts in Technological Context

While evaluating directions for future research in an area such as SQA, it is important to assess the technology context that was considered while the body of knowledge evolved and compare the same to changes that are taking place. We consider it useful to point out two dramatic shifts in technological context that are redefining software delivery in organizations and therefore should have substantial implications for the SQA domain. These are rapid advances in development process technology and client/server distributed computing.

In general, several new system development approaches have been introduced in the last few years. These include CASE tools, integrated methodologies such as information engineering, and object-oriented prototyping approaches. These new process technologies are radically different from traditional methods and several companies have failed in the implementation of these approaches. However, the proponents of each approach claim that the technology can significantly redefine software quality and development productivity. There is little empirical research examining the actual impact of these approaches on specific quality and productivity metrics. Of course, the emergence of new process technology calls for a careful reexamination of the appropriateness and completeness of the software characteristics considered. It is conceivable that a finer grained specification of certain characteristics may be warranted in certain methodology contexts.

Future research should consider the contingencies associated with variations in software process technology from a technical, managerial, organizational and economic standpoint. While some commonalities may exist, some quality assurance issues associated with different approaches should conceivably be different. The problem is further accentuated with a radical shift taking place in organizational computing as firms go from centralized mainframe approaches to distributed computing infrastructures. Here again, there is a clear need to carefully reassess the software characteristics associated with a radically different environment.

While distributed systems are being conceptualized as an enabling technology to transform organizations and dismantle hierarchical structures, the promise assumes that development methods will be able to deliver software that possesses key attributes such as reliability and maintainability. The vendor community has attempted to respond to the challenge with the introduction of tools such as client/server based CASE and object-oriented products. As per our earlier discussion on contingencies associated with process technology, it is necessary that researchers consider the contingencies associated with distributed environments.

8.5. The Conceptual Orientation of SQA Research

A comment is warranted about the conceptual orientation of SQA research. SQA programs and their successful implementation have been modeled as rational decisions made by managers. Their impacts, in turn, have largely been studied from a rationalistic perspective. The IS implementation area adopted a similar approach in the early '80s. However, over time the IS implementation area (as with the organizational innovation area) was characterized by fragmented and contradictory results. The term "subtheories" is normally used while discussing present theoretical understanding in these areas to emphasize the lack of a theory (Damanpour, 1991).

It is not our purpose to take a position against rational models. On the contrary, evidence from the IS implementation area suggests that such models are useful to model adoption behavior but they possess little power in explaining post-adoption behavior. In a recent study investigating the uptake of I-CASE technology, Orlikowski and Robey (1991) emphasize the importance of studying the social context into which the technology is being introduced. A process based approach that embraces a social-interaction perspective can provide useful in-

sights into the dynamics associated with the effective implementation of an SQA program.

8.6. CONCLUDING REMARKS

We conclude by making the following broad observations:

- There is a need for continuity of research effort and systematic studies in a specific area. The series of studies on localized quality metrics by Poore (1988), Binder and Poore (1990), and Trammel and Poore (1992) are good examples of an approach that should benefit the field tremendously.
- Rapid technological changes call for a careful reassessment of software characteristics, metrics, managerial and organizational issues.
- Changes in technology may obliterate existing understanding regarding quality and productivity. A good example is reusability which clearly questions lines of code and function-point based measures to assess productivity.
- The technological context should be an integral part of SQA research. Quality assurance in radically different technological contexts may call for substantially different approaches. The study of the interaction between technological and organizational context is clearly important and presently overlooked.
- Some critical areas have received little or no research attention as yet. A good example is the study of efficiency from a quality perspective.
- Interrelationships between dimensions have been minimally investigated. Given the interdisciplinary thrust of the SQA domain, key thematic bridges between the (sub)dimensions need to be further developed. These thematic bridges will in turn define the knowledge structure of the SQA domain.
- Future research should examine how quality can be engineered into the development process. Some research in this direction has been initiated.
- There is a scarcity of empirical research and only one longitudinal study was identified.
- A social interaction perspective may prove useful in understanding the dynamics of implementing SQA programs or redefining development process technology. This understanding should facilitate better management of implementation efforts during organizational change.
- As the field evolves, the classification system employed here should be reevaluated.

ACKNOWLEDGMENT

The authors would like to thank the Pontikes Center for the Management of Information at Southern Illinois University at Carbondale for partially supporting this project.

REFERENCES

Abdel-Hamid, T., The Economics of Software Quality Assurance: A Simulation-Based Case Study. *MIS Quarterly*, 12, 395–411 (1988).

Ahituv, N. and M. Zelek, Instant Quality Control of Large Batch Processing Jobs. *MIS Quarterly*, 11, 313–323 (1987).

Alavi, M., An Assessment of the Prototyping Approach to Information Systems Development. *Communications of the ACM*, 27, 556–563 (1984).

Allot, K., The Weak Link in the Safety Chain. *Process Engineering*, 72, 41–42, (1992).

Andersen, O., Industrial Applications of Software Measurements. *Information and Software Technology*, 34, 681–693, (1992).

Apte, U., Sanker, C., Thakur, M. and J. Turner, Reusability-Based Strategy for Development of Information Systems: Implementation Experience of a Bank. *MIS Quarterly*, 14, 421–433 (1990).

Banker, R. and R. Kauffman, Reuse and Productivity in Integrated Computer-Aided Software Engineering: An Empirical Study. *MIS Quarterly*, 15, 375–401 (1991).

Barnes, B. and T. Bollinger, Making Reuse Cost-Effective. *IEEE Software*, 8, 13–24 (1991).

Basili, V. and H. Rombach, Implementing Quantitative SQA: A Practical Model. *IEEE Software*, 4, 6–9 (1987).

Bendifallah, S. and W. Scacchi, Understanding Software Maintenance Work. *IEEE Trans. on Software Engineering*, 13, 311–323 (1987).

Bhide, S., Generalized Software Process-Integrated Metrics Framework. *J. of Systems and Software*, 12, 249–254 (1990).

Biggerstaff, T. and C. Richter, Reusability Framework, Assessment, and Directions. *IEEE Software*, 4, 41–49 (1987).

Binder, L. and J. Poore, Field Experiments with Local Software Quality Metrics. *Software: Practice and Experience*, 20, 631–647 (1990).

Boehm, B., *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

Boehm, B., Brown, J., Kasper, H., Lipow, M., MacLeod, G. and M. Merrit, *Characteristics of Software Quality*, North-Holland Publishing Company, Amsterdam, Holland, 1978.

Boehm, B. and R. Ross, Theory-W Software Project Management: Principles and Examples. *IEEE Trans. on Software Engineering*, SE-15, 902–916 (1989).

Brelsford, J., Establishing a Software Quality Program. *Quality Progress*, 21, 34-37 (1988).

Buckley, F. and R. Poston, Software Quality Assurance. *IEEE Trans. on Software Engineering*, SE-10, 36-41 (1984).

Camuffo, M., Maiocchi, M. and M. Morselli, Automatic Software Test Generation. *Information and Software Technology*, 32, 337-346 (1990).

Carpenter, M. and H. Hallman, Quality Emphasis at IBM's Software Engineering Institute. *IBM Systems J.*, 24, 121-133 (1985).

Carpenter, C. and G. Murine, Measuring Software Product Quality. *Quality Progress*, 17, 16-20 (1984).

Carrol, J. and J. McKendrea, Interface Design Issues for Advice-Giving Expert Systems. *Communications of the ACM*, 30, 14-31 (1987).

Cashman, P. and A. Holt, A Communication-Oriented Approach to Structuring the Software Maintenance Environment. *Software Engineering Notes*, 5, 4-17 (1980).

Cerveny, R., Garrity, E. and G. Sandes, Prototyping in Systems Development. *J. of Management Information Systems*, 3-2, 52-62 (1986).

Cho, C., *Quality Programming: Developing & Testing Software with Statistical Quality Control*, John Wiley & Sons, Inc., New York, New York, 1987.

Collofello, J. and J. Buck, Software Quality Assurance for Maintenance, *IEEE Software*, 5, 46-51 (1987).

Cooprider, J. and J. Henderson, Technology-Process Fit: Perspectives on Achieving Prototyping Effectiveness, *J. of Management Information Systems*, 7-3, 67-87 (1990).

Crosby, P., *Quality is Free: The Art of Making Quality Certain*, New American Library, New York, New York, 1980.

Cusumano, M., The Software Factory: A Historical Interpretation. *IEEE Software*, 6, 23-30 (1989).

Cusumano., M. and C. Kemerer, A Quantitative Analysis of U.S. and Japanese Practice and Performance in Software Development, *Management Science*, 36, 1384-1406 (1990).

Dagwell, R. and R. Weber, Systems Designers' User Models: A Comparative Study and Methodological Critique. *Communications of the ACM*, 26, 987-997 (1983).

Damanpour, F., Organizational Innovation: A Meta-Analysis of Effects of Determinants and Moderators. *Academy of Management J.*, 34, 555-590 (1991).

Daskalantonakis, M., A Practical View of Software Measurement and Implementation Experiences within Motorola. *IEEE Trans. on Software Engineering*, 18, 998-1010 (1992).

Dickson, G. and DeSanctis, The Management of Information Systems: Research Status and Themes, in *Research Issues in IS: An Agenda for the 1990s*, (M. Jenkins, H. Siegle, W. Wojtkowski and G.Wojtkowski, eds.), Wm. C. Brown, Dubuque, Iowa, 1990.

Dos Santos, B., Justifying Investments in New Information Technologies. *J. of Management Information Systems*, 7-4, 91-106 (1991).

Dos Santos, B., Peffers, K. and D. Mauer, The Impact of Information Technology Investment Announcements on the Market Value of the Firm. *Information Systems Research*, 4, 1-23 (1993).

Edwards, C., Information Systems Maintenance: An Integrated Perspective. *MIS Quarterly*, 8, 237-256 (1984).

Farbey, B., Software Quality Metrics: Considerations about Requirements and Requirement Specifications, *Information and Software Technology*, 32, 60-64 (1990).

Feinawer, L., Compiler Issues Associated with Safety-Related Software. *Nuclear Technology*, 93, 116-122 (1991).

Franz, C., User Leadership in Systems Development Life Cycle. *J. of Management Information Systems*, 2-2, 5-25 (1985).

Gaffrey, J. and T. Durek, Software Reuse—Key to Enhanced Productivity: Some Quantitative Models. *Information and Software Technology*, 31, 258-267 (1989).

Gill, G. and C. Kemerer, Cyclomatic Complexity Density and Software Maintenance Productivity. *IEEE Trans. on Software Engineering*, 17, 1284-1288 (1991).

Gould, J. and C. Lewis, Designing for Usability: Key Principles and What Designers Think. *Communications of the ACM*, 28, 300-310 (1985).

Grady, R., Measuring and Managing Software Maintenance. *IEEE Software*, 4, 35-45 (1987).

Grumen, G., How to Assure Quality: Debate Shows Divisions. *IEEE Software*, 8, 99+ (1991).

Halasz, F., Reflections on Note Cards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM*, 31, 836-852 (1988).

Hallonan, D., Manches, S., Moriarty, J., Riley, R., Rohrman, J. and T. Skramstad, Systems Development Quality Control. *MIS Quarterly*, 2, 1-14 (1978).

Hansen, M., Software: The New Frontier in Safety. *Professional Safety*, 35, 20-23 (1990).

Harrison, W., Using Software Metrics to Allocate Testing Resources. *J. of Management Information Systems*, 4-4, 93-105 (1988).

Heitkoetter, U., Helling, B., Nolte, H. and M. Kelly, Design Metrics and Aids to Their Automatics Collection. *Information and Software Technology*, 32, 79-87 (1990).

Henderson, J. and J. Cooprider, Dimensions of I/S Planning and Design Aids: A Functional Model of CASE Technology. *Information Systems Research*, 1, 227-254 (1990).

Henderson, J. and S. Lee, Managing I/S Design Teams: A Control Theories Perspective. *Management Science*, 38, 757-777 (1992).

Hollocken, C., Finding the Cost of Software Quality. *IEEE Trans. on Engineering Management*, 33, 223-228 (1986).

Humphrey, W., *Managing the Software Process*, Addison-Wesley, Reading, MA, 1989.

Hunter, R., Where Next in Software Standards. *Software Quality J.*, 1, 1-8 (1992).

Ince, D., Software Metrics: Introduction. *Information and Software Technology*, 32, 297-303 (1990).

Kane, E., Implementing TQM at Dun & Bradstreet Software. *National Productivity Review*, 405-416 (1992).

Karimi, J., An Asset-Based Systems Development Approach to Software Reusability. *MIS Quarterly*, 14, 179-198 (1990).

Kemerer, C., An Empirical Validation of Software Cost Estimation Models. *Communications of the ACM*, 30, 416-432 (1987).

Kim, C. and S. Westin, Software Maintainability: Perceptions of EDP Professionals. *MIS Quarterly*, 12, 167-179 (1988).

Kishida, K., Teramoto, M., Torii, K. and Y. Urano, Quality-Assurance Technology in Japan. *IEEE Software*, 4, 11-17 (1987).

Kitchenham, B. and S. Linkman, Design Metrics in Practice. *Information and Software Technology*, 32, 304-310 (1990).

Kuster, R., van Genuchten, M. and F. Heemstra, Are Software Cost-Estimation Models Accurate? *Information and Software Technology*, 32, 187-190 (1990).

Levendel, Y., Reliability Analysis of Large Software Systems: Defect Data Modeling. *IEEE Trans. on Software Engineering*, 16, 141-152 (1990).

Levendel, Y., Improving Quality with a Manufacturing Process. *IEEE Software*, 8, 13-25 (1991).

Leveson, N., Cha, S. and T. Shimeall, Safety Verification of Ada Programs Using Software Fault Trees. *IEEE Software*, 8, 48-59 (1991).

Lucas, H. and N. Nielson, The Impact of the Mode of Information Presentation on Learning and Performance. *Management Science*, 26, 982-993 (1980).

Mantei, M. and T. Teorey, Cost/Benefit Analysis for Incorporating Human Factors in the Software Life Cycle. *Communications of the ACM*, 31, 428-439 (1988).

McCabe, T. and C. Butler, Design Complexity Measurement and Testing. *Communications of the ACM*, 32, 1415-1425 (1989).

Miller, H., Quality Software: The Future of Information Technology. *J. of Systems Management*, 12, 8-14 (1989).

Mukhopadhyay, T., Vicinanza, S. and M. Prietula, Examining the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation. *MIS Quarterly*, 16, 155-171 (1992).

Munson, J. and T. Khoshgoftaar, The Detection of Fault-Prone Programs. *IEEE Trans. on Software Engineering*, 18, 423-433 (1992).

Murine, G., Integrating Software Quality Metrics with Software Quality Assurance. *Quality Progress*, 21, 38-41 (1988).

Musa, J. and W. Everett, Software Reliability Engineering: Technology for the 1990s. *IEEE Software*, 7, 36-43 (1990).

Napier, H., Lane, D., Batsell, R. and N. Guadango, The Impact of a Restricted Natural Language Interface on Ease of Learning and Productivity. *Communications of the ACM*, 32, 1190-1198 (1989).

Nenz J., Software Quality Assurance: Systems 12. *Electrical Communication*, 59, 68-73 (1985).

Newman, M. and D. Robey, A Social Process Model of User-Analyst Relationships. *MIS Quarterly*, 16, 249-266 (1992).

Niederman, F., Brancheau, J. and J. Wetherbe, Information Systems Management Issues for the 1990s, *MIS Quarterly*, 15, 475-500 (1991).

Okumoto, K., A Statistical Method for Software Quality Control. *IEEE Trans. on Software Engineering*, 11, 1424-1430 (1985).

Orlikowski, W. and D. Robey, Information Technology and the Structuring of Organizations. *Information Systems Research*, 2, 143-169 (1991).

Paughtrey, T., The Search for Software Quality. *Quality Progress*, 21, 29-31 (1988).

Pfleeger, S., Lessons Learned in Building a Corporate Metrics Program. *IEEE Software*, 10, 67-74 (1993).

Pfleeger, S. and T. Bollinger, The Economics of Reuse: New Approaches to Modeling and Assessing Cost. *Information and Software Technology*, 36, 475-484 (1994).

Pfleeger, S. and C. McGown, Software Metrics in the Process Maturity Framework. *J. of Systems and Software*, 12, 255-263 (1990).

Poore, J., Derivation of Local Software Quality Metrics (Software Quality Circles). *Software: Practice and Experience*, 18, 1017-1027 (1988).

Poston, R. and M. Bruen, Counting Down to Zero Software Failures. *IEEE Software*, 4, 54-61 (1987).

Prieto-Diaz, R. and P. Freeman, Classifying Software for Reusability. *IEEE Software*, 4, 6-16 (1987).

Rai, A., Stubbart, C. and D. Paper, Can Executive Information Systems Reinforce Biases? *Accting., Mgmt. & Info. Tech.*, 4, 87-106 (1994).

Rahman, W., Software Quality by Management: Learning from the Manufacturing Industries. *Information and Software Technology*, 29, 511-516 (1987).

Ramamoorthy, C. and F. Bastani, Software Reliability—Status and Perspectives. *IEEE Trans. on Software Engineering*, 8, 354-371 (1982).

Reynolds, R., The Partial Metrics System: Modeling the Stepwise Refinement Process Using Partial Metrics. *Communications of the ACM*, 30, 956-963 (1987).

Rivard, E. and K. Kaiser, The Benefit of Quality IS. *Datamation*, 35, 53-58 (1989).

Rombach, A., A Controlled Experiment on the Impact of Software Structure on Maintainability. *IEEE Trans. on Software Engineering*, 13, 344-354 (1987).

Rush, K., Draving, S. and J. Kerley, Technical Challenges to a Decentralized Phone System. *IEEE Spectrum*, 21, 32-37 (1990).

Schneidwind, N., Methodology for Validating Software Metrics. *IEEE Trans. on Software Engineering*, 20, 410-422 (1992).

Schneidwind, N., The State of Software Maintenance. *IEEE Trans. on Software Engineering*, 15, 303-310 (1987).

Siegel, S., Why We Need Checks and Balances to Assure Quality. *IEEE Software*, 9, 102–103 (1992).

Siegrist, K., Reliability of System with Markov Transfer of Control, II. *IEEE Trans. on Software Engineering*, 14, 1478–1480 (1988).

Swanson, K., McComb, D., Smith, J. and D. McCubbrey, The Application Software Factory: Applying Total Quality Techniques to Systems Development. *MIS Quarterly*, 15, 567–579 (1991).

Tait, P. and I. Vessey, The Effect of User Involvement on System Success: A Contingency Approach. *MIS Quarterly*, 12, 91–108 (1988).

Tenny, T., Program Readability: Procedures Versus Comments. *IEEE Trans. on Software Engineering*, 14, 1271–1279 (1988).

Trammel, C. and J. Poore, A Group Process for Defining Local Software Quality: Field Applications and Validation Experiments. *Software: Practice and Experience*, 22, 603–636 (1992).

Van Treeck, G. and R. Thackeray, Quality Function Deployment at Digital Equipment Corporation. *Concurrent Engineering*, 1, 14–20 (1991).

Vessey, I. and R. Weber, Some Factors Affecting Program Repair Maintenance: An Empirical Study. *Communications of the ACM*, 26, 128–134 (1983).

Weill, P., The Relationship Between Investment in Information Technology and Firm Performance: A Study of the Valve Manufacturing Sector. *Information Systems Research*, 3, 307–333 (1992).

Wilden, J., Wolft, A., Rosenblatt, W. and P. Tarr, Specification-Level Interoperability. *Communications of the ACM*, 34, 72–87 (1991).

Yu, W., Smith, D. and S. Huang, Software Productivity Measurement. *AT & T Technical J.*, 69, 110–120 (1990).

Zahedi, F., Reliability of Information Systems Based on the Critical Success Factors—Formulation. *MIS Quarterly*, 11, 187–203 (1987).

Zulter, R., The Deming Approach to Software Quality Engineering. *Quality Progress*, 21, 58–64, (1988).