An Exploratory Analysis Of Issues In System & Acceptance Testing

James A. Spruell, (James.Spruell@Rockhurst.Edu), Rockhurst University
Ricard E. Downing, (Ricard.Downing@Rockhurst.Edu), Rockhurst University

ABSTRACT

This paper uses findings from a qualitative study to explore the issues identified by programmers and the end user community in conducting software testing. Narratives were collected from 103 "software testers" and project managers selected from 10 different organizations in the Kansas City/Jefferson City area. ***Semi-structured interviews*** *were used to illuminate the background to, and ways in which people conceptualize and respond to testing issues in their specific organizational setting. The study findings suggest a number of managerial strategies to ensure a more efficient testing process.*

INTRODUCTION

In today's competitive climate organizations have become increasingly dependent upon software to perform critical operations. The organization has evolved into a system of information technology that is intertwined with a collage of individuals and capital that work together to accomplish day-to-day activities. In this environment, the importance of fault tolerant, error free information systems has become crucial to the organization's competitive existence.

The increase in the automation of services, transactional processing, the analysis of organizational data, etc. has left many organizations very vulnerable to service interruptions (Peterson, 1991; Dalal and Mallows, 1998). The software error that disrupted AT&T's long-distance service in 1990 cost the organization more than $60 million in lost revenues (Peterson, 1991). Similarly a failure in the order entry, accounts receivable, or inventory control systems in a large organization could have potentially disastrous effects.

As important as it is, the software industry has not done a very good job of producing error free / fault tolerant software. Significant customer dissatisfaction with software is well reflected in the results of a 1996 survey that found over 200 million calls for technical support at an average cost of $23 per call. These customers were left on hold for approximately 3 billion minutes, worse than any other industry (Kaner et al., 1993; Kaner & Pels, 1997; Kaner, 1999). Testing along with quality assurance has been identified as a critical activity in implementing business applications and in detecting and preventing the release of error-ridden software (Mayer, 1998).

Project Rationale

A number of software testing studies have examined the quantitative aspects of testing the statistics and the automation of test processes. Many focus on validating testing models, when to stop testing, or estimation techniques. (Dalal and Mallows 1988; Singpurwalla 1991; Dalal and McIntosh 1994; Dalal and Mallows 1998). Other studies focus on the generation of test cases and the automation of test processes (Dalal and Patton, 1993; Burrough et al., 1994; Burr and Young, 1997; Cohen et. al., 1997; Dalal and Mallows, 1998). Kan has emphasized the importance of in-process measurements in tracking the progress and management of testing (Kan, 2001).

Dalal and Mallows (1998) suggested that three approaches to improving testing productivity lie in the more efficient generation of test cases, the automation of the test process and the management of testing. The Content-

Driven School of Testing (Petticord, 2003) further supports testing as a management or people activity by proposing that people create software, and that people set the context. Kaner (2004) extends this concept to hypothesize that software testing is a social science in which human factors may dominate. This approach emphasizes that testing operates under uncertainty, and for a given product testers must learn or identify the project goals, desired outcomes, and the possible consequences of failure. The learning component of testing becomes even more important when the complexity of software testing as well as the information system is considered.

The more rigorous approaches to systems and acceptance testing are knowledge intensive and may include a combination of rules, guidelines, methodologies, techniques, and software testing tools. The transfer of this knowledge, or at least a minimal base that would allow IT personnel and end users to effectively conduct system and acceptance testing, can be problematic. Firstly, the knowledge transfer is asymmetric between the Information Test Group and the end users selected for testing. IT personnel would possess the technical knowledge with end users possessing the procedural and business acumen. This asymmetry along with motivational differences between participants would create analogous barriers to knowledge transfer described by Ko, et. Al (2005).

Systems acceptance testing also occurs late in the development life cycle limiting the time available to make adequate corrections. Additionally, on a large system many programmers or analysts that have completed their specific tasks may have been assigned to other projects thus reducing the resources essential to making corrections. The constraints of limited time, resources, and the specific knowledge necessary can combine to create potentially arduous relationships between participants.

Rather than focus on the analytical aspects of testing, this study examines the human factors of planning and conducting testing. By using a qualitative study, insight was gained into testing issues that the more quantitative studies would have missed. Conclusions drawn from the narratives help explain the concerns of both the novice and experienced tester, and provide substantial insight into the realities of testing. A variety of options were suggested for managerial action to improve upon the testing process.

The research study attempted to address two primary objectives:

- The first is to explore the issues that software testers face in conducting testing in an organizational setting, an area with few previous studies.
- The second is to understand the differences in perspectives between the end-user community and developers who produced the software.

THE STUDY

Semi-structured interviews were considered the most appropriate means of identifying the key issues that programmers and end users face in conducting software testing. To this end, data were collected from 103 individuals in the Midwest region that either had been selected by their respective management as software testers, or were serving as project managers/ full time testers, for a variety of major IT projects.

In the first stage, an initial convenience sample of 6 project managers and 7 fulltime software testers was identified. Each individual was contacted and provided a brief description of the research. To collect the data, face-to-face or telephone interviews were conducted with each of the respondents.

Semi-structured interviews formed the basis of the data collection in this first stage with each participant allowed to provide an open ended response to what they believed were the most significant issues involved in conducting system and acceptance testing, which issues they believed were the most important, and an explanation of the responses.

In the second stage an additional 90 individuals were identified who had been selected by their respective project managers to participate in systems and acceptance testing for one or more major system rollouts. Each participant was asked to provide a written response to the same questions asked in the first stage. The written responses were followed up with group and individual interviews to provide a greater understanding of responses.

This sample included individuals who typically had little formal software testing experience, and represented persons selected to participate in unit, system and acceptance testing. In many cases, the sample included both the developers of the system and members of the end-user community. Interviews were conducted from June of 1997 to December of 2003 in a longitudinal study reflecting major system implementations during the study period.

The study's inclusion of multiple organizations, as well as sub-groups within the organization, provided a richer understanding of the many facets of software testing. The diversity also provides a measure of protection against the bias generated with mono-method survey approaches (Crampton and Wagner, 1994; Soon and Slaughter, 2001)

Interview narratives were substantiated with a variety of internal and external documents. Sample test plans, output from the problem reporting systems, and test folders were collected along with documents that identified test standards and procedures. Combining these documents with interview narratives provided both additional insight and additional rigor to the study (Eisenright, 1989)

RESULTS AND DISCUSSION

The Different Perspectives

The *interviews* yielded multiple perspectives from the different sub groups selected for the study. We classified study participants on the basis of their previous experience in testing as either new testers or experienced testers. An experienced tester was one who had participated in at least one systems/acceptance testing involved in a major system roll out. The group was then further divided into those from the user community and project developers recruited (assigned) to testing. The resulting four quadrants are shown in figure 1 along with the percentage of respondents in each area. A fifth group of 13 individuals was interviewed who served as (7) full time testers or (6) project managers. The full time testers, a small but important section, were assigned to their organization's equivalent of the Information Test Group.

Figure 1: Grouping Of Testers Based On Their Test Experience And Assignment As An End User Or A Software Developer (Figure does not include project managers or those assigned as full time testers (ITG))

	New Testers	**Experienced Testers**
End Users	Quadrant I (26 %)	Quadrant 2 (11 %)
Programmers	Quadrant 3 (21 %)	Quadrant 4 (35 %)

A comparison of study participants, their responses, and approaches revealed some marked contrasts between quadrants. The general education level, absorptive capacity of information receivers, language constructs, and even differences in gender of end users differed from their counterparts. New Testers were more likely to have less formal education, use non-technical language, be female, and serve in largely clerical roles.

The variation in background, approaches and perspectives often reflected important differences in factors that contribute to knowledge transfer as well as complexity. A partial summary of those variations is presented below

Complexity. Complexity is a human construct that is perhaps best defined relative to the individual. In examining the tasks, knowledge domain, and elements faced by participants variation was evident in how complex an item/element was relative to the participant and included:

- Software testing as a process
- The actual information system being implemented
- Procedural items, i.e., the complexity of the business tasks being performed
- Elements of the business domain that related to the specific functional area of the business, e.g., a health care system for reporting bacterial culture results.

For an end user who had never participated in testing, the actual testing process was relatively complex. This complexity was reduced as a participant gained in subsequent training and involvement with one or more system roll-outs.

Comments from the IT participants could be traced back to the complexity of the business procedures and functional expertise with which they might not be familiar.

Knowledge Transfer. Performing systems and acceptance testing potentially involves a substantial knowledge transfer between the Information Test Group and end users selected for testing. Ko, et.al., (2005) previously identified several factors that impact the knowledge transfer from consultant to client in implementing complex ERP information systems. These included:

- Potential for arduous relationships between personnel;
- Recipient's intrinsic motivation levels;
- Source's intrinsic motivation levels;
- Source credibility;
- Shared understanding of participants;
- Encoding and decoding abilities.

When participants from each of the four quadrants were compared against these factors key differences surfaced. For example, the motivation to perform testing had a different basis for IT personnel vs. end users. The assignment as an end user tester was typically made by managers, temporary, and had little impact on career advancement. The outcome of the testing though would impact the quality of the system that would be used at some future date. The discovery of substantial flaws could conceivably impact the career of IT personnel.

The potential for an arduous relationship also existed between end users and IT professionals. Several members of Quadrant III expressed negative comments about working with end users involved in testing. On the other hand, end users rarely made negative comments toward IT personnel, and indeed often had little prior experience with IT. This created the potential for an asymmetric arduous relationship with uneven levels of animosity among the quadrants.

Testers who had never participated in prior testing may have lacked the experience to fully appreciate the role of systems and acceptance testing thus creating a lack of shared understanding. Inexperience with the language and terms common to testing appeared to also contribute to this state.

A similar lack of shared understanding existed between quadrants on the role and purpose of the proposed systems. Clerical people for example understood that small design flaws would substantially increase the time required to fill out forms, etc. while IT personnel often viewed such situations as end user preferences.

Learning is a process that involves communication, and the ability to encode/decode business procedures, domain, and information system constructs. Substantial differences were apparent in ability and experience level in encoding/decoding information.

Quadrant 1 (New End User Testers)

This group took their role as testers quite seriously, and accepted the break from their normal assignments well. The critical issue among new testers from the end-user community was relatively predictable… will I receive adequate training, when do I know when I'm done, and what do you want me to do.

Quadrant 2 (Experienced End User Testers)

The issues changed for end users who had been involved previously in system and acceptance testing in at least one major roll out. Members of this quadrant were much more concerned about issues involving management support and problem resolution. Various individuals expressed concern about past experiences where they felt they had made a good effort, but that often problems went unresolved, especially in the areas of design and usability.

Remarkably, the group was still very interested in their role as a tester and took it quite seriously. A striking difference between experienced IS testers and the end user community was that the end users were much less concerned about schedule issues with almost no one mentioning the lack of time. One possible explanation is that they didn't come into testing with a history of projects behind schedule. The individuals seamed generally content with the time allotted.

Quadrant 3 (New IS Testers)

A few disgruntled comments surfaced among new testers from the IS side. Such comments took the form of "*Customers are not familiar with the request and waste programmer's time with bogus questions or issues*" or simply "*Customers don't know what they want.*" The comments appeared to be largely a result of prior end-user/programmer analyst interaction. These individuals tended to classify design errors as user requests, bugs as user errors, etc.

Many of these programmers had been involved only in unit testing, and had never been through the acceptance testing process. In two cases the pessimism disappeared after a 3-day side-by side testing training workshop involving end users and programmers.

This group also expressed concern about the basics of testing as the new end-user testers, with subtle differences in how the concerns were expressed. For example, one individual phrased his concerns as "*Who* decides when we stop?" vs. "How do I know when to stop?"

Quadrant 4 (Experienced IS Testers)

This quadrant was by far the largest and most vocal of the four quadrants. For this quadrant, the most important issue was that of scheduling, followed by testing standards and issues that involved the "test climate". A breakdown of these items is displayed in Table 1.

Table 1: The Top Issues Identified By Experienced IS Testers

Quadrant 4 Issues

Scheduling:

- Not enough time to conduct quality testing. (52%)

Testing Standards:

- No [or inadequate] testing standards. (28%)
- No enforcement of standards. (21 %)
- No structure or plan to testing. (25%)

Test Climate:

- Lack of management support for testing. (38%)
- Inadequate training. (28%)
- Lack of communication between users, DP, & management. (28%)

Scheduling was the number one issue identified by Quadrant 4 testers. The result is not surprising although the concern was not just time, but having enough time to do a quality job. One individual described the problem as "Deadlines *are usually placed too near the start date [of testing], and are too inflexible to ensure adequate testing.*" A second person placed the emphasis on "*Resolving any problem reports timely [in a timely manner].*" Clearly this group had struggled with the issues of performing testing, identifying problems, resolving those problems, and retesting.

This problem appeared to be aggravated in several of the organizations by the scheduling changes during the course of testing. One account reflected this as "*Changes [are] given constantly during testing even after something is supposed to be in production making the project late.*" Another stated similar concerns as *"Trying to remain flexible with changes to the original game plan keep peppering in."*

Many of issues centered on the concept of a deadline. The experience of Quadrant 4 testers would suggest that instead of setting the goal to have the system in production by… to have the goal to move the system to testing by… with a percentage of total development time allocated to testing.

A second major area was the importance of having testing standards in place, and enforcing those standards. Over 25 per cent of this group also mentioned that they felt that there was little testing structure or planning.

Among experience testers many of the issues centered on managing the testing process and reducing ambiguity. The comments about the lack of standards and their enforcement, inadequate training, lack of management support, and communication issues all seem to have a similar root - the need to reduce ambiguity faced by a set of highly structured individuals.

In reviewing the documents, test plans and problem-reporting procedures appeared to be in place. Hence, the real issue may be in communicating and enforcing procedures, and standards across the various testing teams. The lack of communication and management support elevated the ambiguity rather than reduced it.

Two minor areas frequently mentioned by experienced testers were communication and understanding requirements. The importance of these two issues is perhaps a reflection of the complexity of automating business functions. David Parnas speculated that software errors are not so much a result of a lack of knowledge but are "blunders caused by our inability to fully understand the intricacies of these complex products" (Peterson, 1991, p. 86)

Information Test Group/Project Managers

Individuals from the various Information Test Groups were the most global in their interests and often focused more on methodologies than specific procedures. The need for better software tracking software, consistent department wide test plans and guidelines, established methodologies, etc. were mentioned. Other examples that also reflected the more conceptual nature of their concerns included comments such as:

- *"What percentage will constitute a fatal, serious or minor error?"*
- *"When do you give up on an issue?"*
- *"Does the program follow all state rules and regulations?"*

Project managers often (repeatedly) mentioned the difficulty in establishing what was a "user error" vs. a design flaw. Dialogs on this topic between experienced testers and end-users were oft cited along with the difficulty of convincing each group of accepted guidelines herein.

Overview

None of the issues concerning when to stop testing were embedded in statistics. Noticeably missing even among members of the various Information Test Groups was *any* mention/concern of regression testing or statistical analysis to determine if adequate coverage had occurred. Several testers assigned to ITGs were quite conversant about regression testing but were not concerned about the use of statistical methodologies.

We believe this behavior is a reflection of the reality faced by testers of large system rollouts. System and acceptance testing involves a collage of testers that often includes members from each of the four quadrants along with ITG. With tight schedules, the need to coordinate between many diverse groups on a large project, the realities of testing is that testing stopped when time ran out, or an experienced tester/project manager declared it was time to stop. This supports Kaner's conclusion that many aspects of testing are largely heuristic, and that "extent of testing metrics" are judgment-driven.

We also can safely say that in the groups of testers that we examined, decisions regarding testing were based on a "satisficing" decision process vs. any maxi-max, or optimizing procedure. The tools, knowledge among a diverse group, and scheduling for a large project simply does not allow exhaustive enumeration of when to stop or how much coverage is required. Future research is needed to verify if testers are minimizing the maximum regret for their decisions, or simply using a satisficing approach, where intuitive judgment says... this meets the general expectations and its time to stop.

Modeling a Solution

The concerns across the different quadrants were restated as an organizational testing "Self Evaluation List" that is presented in Table 2. With these items in place, an organizational model of systems and acceptance testing is shown in Figure 2. The table identifies key concerns expressed by a variety of quadrants, and is viewed as a significant step to resolving many issues.

Table2: A Self-Evaluation List For Systems & Acceptance Testing

System & Acceptance Testing Self Evaluation Check List

Testing Methodology:

1. Testing procedures in place, well documented, and used.
2. A testing plan has been approved.
3. Has decision been made as to what constitutes an error?
4. A bug tracking system is in place, with follow-up.
5. Management supports testing activities with testers having the authority to suspend system implementation.
6. Adequate training for end-user and IS personnel has proceeded.

Scheduling:

1. Adequate time allowed for testing.
2. Change management procedures in place, minimizing additional changes during testing.
3. Flexibility to handle problem resolution, including fixes & retest, ability to suspend project until resolution. (Appeals to executive sponsor).

Motivation:

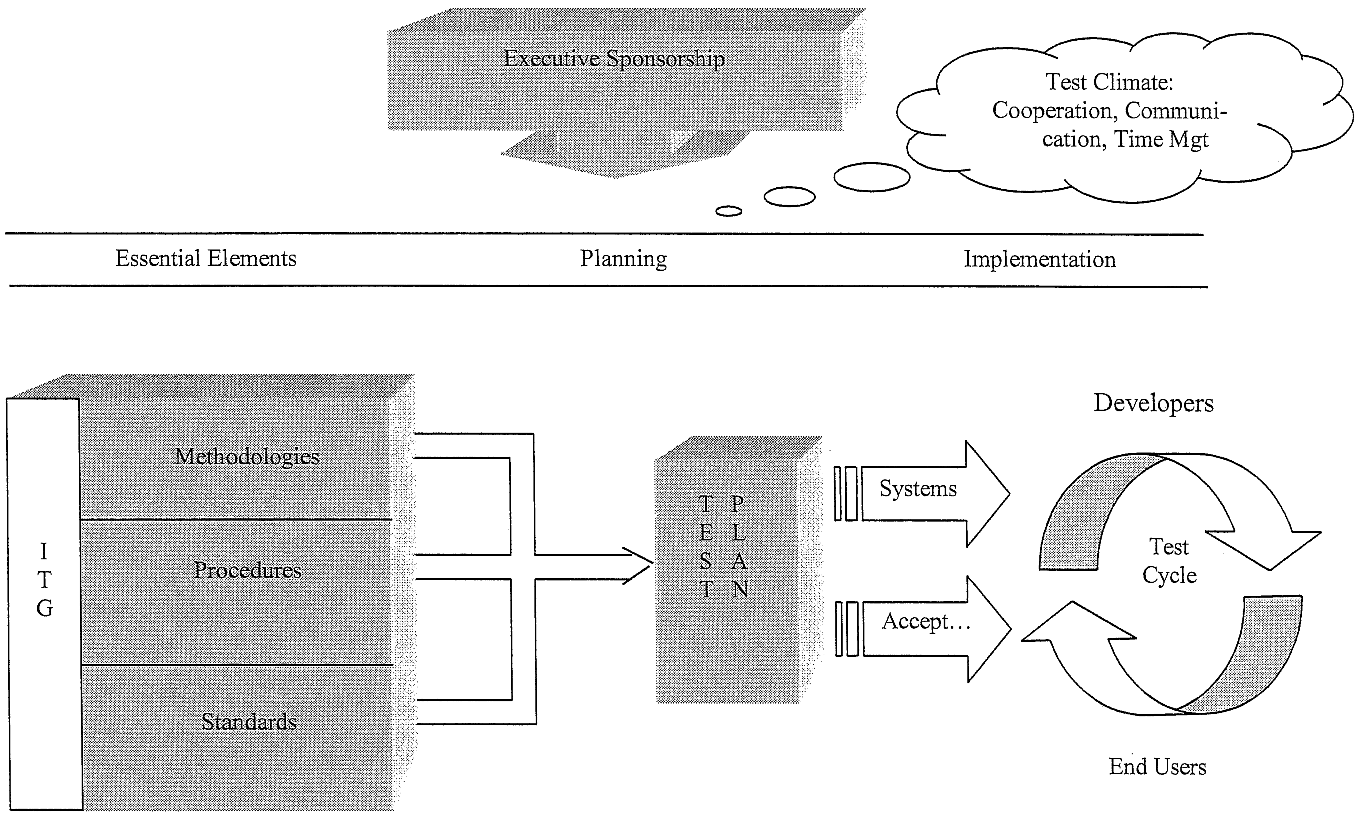
1. Management supports the testing process
2. Testers understand their role and the importance of testing.

Communication :

1. Communication lines are open
2. Communication is occurring between the different groups involved in testing
3. Dialog is open between users/programmers involving interpretations of specifications.
4. Everyone understands the process, procedures, etc.

These fifteen steps accounted for 99 percent of all first round responses. Many cases did not so much mandate a rigorous, optimal set of procedures as having identified and communicated whatever steps were in place. The emphasis is clearly on the support of management for the testing process and enforcement of decisions made along the way. These factors are graphically presented in Figure 2 below.

Figure 2: An Organizational Model Of System And Acceptance Testing



Sample Coverage. First, a significant selection bias may exist in that the study over represented software testing for large-scale information projects at state institutions. Cross validation with data collected from smaller projects, and those at commercial companies, could prove useful.

Furthermore, the source of participants was dependent upon the voluntary cooperation of project managers, and may have excluded useful information from other sources. For example, some corporations would not want to participate in the study due to sensitivity about data that could potentially reflect on the information systems quality.

Reporting Accuracy. The structured interviews employed open-ended questions that collected responses in the participant's language that was often difficult to categorize. The possibility of bias in the interviewer's assessments also exists, and cross validation with quantitative data would add an additional layer of rigor to the study.

Sensitivity Of Data. Organizations are understandably hesitant to advertise potential flaws or weakness in their IT projects particularly in the more visible systems. Hence, data gathered in a study of this type must balance the open candor of research while maintaining anonymity of participating individuals and organizations. This confidentiality limited the study by excluding data from organizations that deemed the information too sensitive. For example in a hypothetical situation a Fortune 50 bank with a very visible cash matching system would want to maintain total confidentiality about issues with their testing efforts. This need also limited the form in which results could be presented to ensure that comments could not be traced back to a particular system or organization. The above findings should be considered in light of the study's limitations

CONCLUSION

Although subjective and interpretive, qualitative analysis did provide substantial insight into the unexplored issues involved in software testing. The knowledge gained holds practical implications for the software industry, and adds substantial relevance as to what goes on in software testing within the organizational settings. We believe that the issues are real, and are experienced by a variety of testers both in the study groups, and in the general population of software testers. The conclusions drawn from the study have a direct contribution to how we conduct software testing, and areas to be emphasized. As Eisenhardt (1989) has pointed out, this approach both increases the credibility and the rigor of a study.

REFERENCES

1. Burr, K. and Young, W. (1997). Test Acceleration (Automatic Efficient Testcase Generations.) Unpublished paper presented at Nortel Design Forum, Ottawa, December 3-5.
2. Burrough, K., Jain, A., and Erickson, R. L. (1994). Improved Quality of Protocol Testing Through Techniques of experimental Design. *Supercomm/ICC'94 (IEEE International Conference on Communications)*, New York: IEEE, pp. 745-752.
3. Cohen, D. M., Dalal, S. R., Fredman, M., and Patton, G. C. (1997). The AETG System: An Approach to Testing Based on Combinatorial Design. *IEEE Transactions of Software Engineering* (23), 437-443.
4. Crampton, S. M. and Wagner III, J. A. (1994). Percept-Percept Inflation in Micro-organizational Research: An Investigation of Prevalence and effect. *Journal of Applied Psychology* (79:1), 67-79.
5. Dalal, S. R. and Mallows, C. L. (1988). When Should One Stop Testing Software? *Journal of the American Statistical Association* (83), 872-879.
6. Dalal, S. R. and Patton, G. C. (1993). Automatic Efficient Test Generator (AETG): A Test Generation System for Screen Testing, Protocol Verification and Feature Interactions Testing. Technical memo, Bllcore, Morristown, NJ as reported in Dalal and Mallows (1998).
7. Dalal, S. R. and McIntosh, A. M. (1994). When to Stop Testing for Large Software Systems with Changing Code. *IEEE Transactions on Software Engineering* (20), 318-323.
8. Dalal, S. R. and Mallows, C. L. (1998). Factor-Covering Designs for Testing Software. *Technometrics* (40:3), 234-244.
9. Eisenhardt, K.M. Building theories from case study research. *Academy of Management Review* (14:4) , pages 532-550.
10. Kan, S. H. (2001). In-Process Metrics for Software Testing. *IBM Systems Journal* (40:1), 220-242.
11. Kaner, C., Falk, J., and Nguyen, H. Q. (1993). *Testing Computer Software*. New York: Van Nostrand Reinhold.
12. Kaner, C. and Pels, D. (1997). Article 2B and Software Customer Dissatisfaction (May 30, 1997) www.badsoftware.com/stats.htm
13. Kaner, C. (1999). Software Engineering and UCITA. *Journal of Computer and Information Law* (18:2), 1-75.
14. Kaner, C. (2004). Software Testing as a Social Science. IFIP Working Group 10.4 meeting on Software Dependability. Siena, Italy
15. Ko, D. Kirsch, L., and King, W. (2005). Antecedents of Knowledge Transfer from Consultants to Clients in Enterprise System Implementation. *MIS Quarterly*, 29, pp. 59-89.
16. Mayer, J. H. The Right Way to View Testing. *Software Magazine,* May 98 Supplement (18:7), 3-4.
17. Peterson, I. (1991). Finding Fault. *Science News* (139: 7), 104- 107.
18. Petticord, B. (2003). Four Schools of Software Testing. Pacific Northwest Software Quality Conference, Portland, Oregon.
19. Singpurwalla, N. D. (1991). Determining an Optimal Time Interval for Testing and Debugging Software. *IEEE Transactions of Software Engineering* (17), 313-319.
20. Soon, A. and Slaughter, S. A. (2001). Work Outcomes and Job Design for Contract Versus Permanent Information systems Professionals on Software Development Teams. *MIS Quarterly* (25:3), 321-347.