Sources of Conflict Between Developers and Testers in Software Development

Xihui Zhang
University of Memphis, xzhang6@una.edu

Jasbir S. Dhaliwal
University of Memphis, jdhaliwl@memphis.edu

Mark L. Gillenson
University of Memphis, mark.gillenson@memphis.edu

Gertrude Moeller
FedEx Express, gmoeller1@fedex.com

Sources of Conflict Between Developers and Testers in Software Development

Xihui Zhang
University of Memphis, xihui.zhang@memphis.edu

Jasbir S. Dhaliwal
University of Memphis, jdhaliwl@memphis.edu

Mark L. Gillenson
University of Memphis, mark.gillenson@memphis.edu

Gertrude Moeller
FedEx Express, gmoeller1@fedex.com

Sources of Conflict between Developers and Testers in Software Development

Xihui Zhang
University of Memphis
xihui.zhang@memphis.edu

Jasbir S. Dhaliwal
University of Memphis
jdhaliwl@memphis.edu

Mark L. Gillenson
University of Memphis
mark.gillenson@memphis.edu

Gertrude Moeller
FedEx Express
gmoeller1@fedex.com

ABSTRACT

Interpersonal conflict between software developers and testers is inevitable and pervasive. This conflict is likely to be negatively associated with software quality and job satisfaction. This study addresses one major research question: What are the sources of interpersonal conflict between developers and testers in software development? Using a qualitative approach, we collect and analyze fifty developer-tester conflict scenarios from professional developers and testers. Preliminary results indicate that conflict sources between software developers and testers fall into three major categories: Process, people, and communication. Conflict sources are presented in a category-subcategory-example format. Implications for research and practice are discussed.

Keywords

Conflict, interpersonal conflict, conflict source, software development, testing.

INTRODUCTION

Software development is a complex process that necessitates interactions between diverse individuals in different roles. These roles typically include end users, business analysts, systems analysts, designers, programmers, testers, and project managers. Interaction can occur between any two roles; and one of the most important interactions in software development process is between developers (a category that typically includes systems analysts, designers, and programmers) and testers (Cohen, Birkin, Garfield, and Webb, 2004; Robey, Welke, and Turk, 2001). One natural outcome of human interaction is interpersonal conflict. Interpersonal conflict results when interdependent parties have different goals, mindsets, values, preferences, backgrounds, and experiences (Barki and Hartwick, 2001; Cohen et al., 2004).

Interpersonal conflict between developers and testers is inevitable and pervasive in software development process, given the inherent task and individual differences between them (Cohen et al., 2004; Simmel, 1964). Prior research has focused on the conflict between end users and IS staff (Beath and Orlikowski, 1994; Ives and Olson, 1984; Robey and Farrow, 1982; Smith and McKeen, 1992; Yeh and Tsai, 2001), and conflict among IS staff (Dos Santos and Hawk, 1988; Wang, Chen, Jiang, and Klein, 2005); however, little research has focused specifically on the conflict between developers and testers.

Prior research indicates that interpersonal conflict between developers and testers is likely to be negatively associated with software quality as well as job satisfaction of the conflicting parties (Barki and Hartwick, 2001; Cohen et al., 2004; Collins, 1986; De Dreu and Weingart, 2003; Jehn, 1995; Wang et al., 2005). Software delivered with poor quality leads to unhappy end users, which in turn, may cause infrequent system use (Hwang and Thorn, 1999), an indicator of system failure (DeLone and McLean, 1992, 2003). Job dissatisfaction is often positively associated with absenteeism, intention to leave, and actual turnover (Hulin, 1990; Jehn, 1995; Jehn, Rupert, and Nauta, 2006; Maslach, Schaufeli, and Leiter, 2001). Given the link between job dissatisfaction and turnover, it is especially important to study developer-tester conflict due to the current and projected shortage of IT talent (Luftman, 2008). It is also critical that IS researchers and practitioners thoroughly understand the sources of developer-tester conflict so that appropriate actions can be taken to mitigate its overall negative impact, which often leads to organizational ineffectiveness and inefficiency (Rahim and Bonoma, 1979).

This research attempts to understand the nature of developer-tester conflict. In particular, it addresses one major research question: What are the sources of interpersonal conflict between developers and testers in software development?

This paper is structured as follows: First, we provide a review of related work. Next, we describe the research methodology. We then present results, concluding with a discussion of findings as well as implications for research and practice.

RELATED WORK

This section reviews previous work related to the sources of interpersonal conflict. It starts with an overview of general literature on interpersonal conflict antecedents, then proceeds to software development related literature, and finally ends with literature related specifically to developer-tester conflict.

Overview of Interpersonal Conflict Antecedents

"Conflict is a process in which one party perceives that its interests are being opposed or negatively affected by another party" (Wall and Callister, 1995, p. 517). Conflict can occur at five different levels, including personal, interpersonal, intergroup, interorganizational, and international (Deutsch, 1990). In this study, we focus on conflict at the interpersonal level, in which an individual is in conflict with other individuals (Wall and Callister, 1995).

Antecedents of interpersonal conflict can be sorted into two broad categories: Individual characteristics and interpersonal factors (Wall and Callister, 1995). Individual characteristics that contribute to interpersonal conflict include personality (Baron, 1989), values (Augsburger, 1992), goals (Coombs and Avrunin, 1988; Pruitt and Rubin, 1986; Wong, Tjosvold, and Lee, 1992), commitment to position (Wall and Callister, 1995), stress and anger (Derr, 1978), and desire for autonomy (Wall and Callister, 1995).

Interpersonal factors known to contribute to interpersonal conflict include perceptual interface, behavior, communications, structure, and previous interactions (Wall and Callister, 1995). Perceptual interface involves belief about another's intentions or motivations, regardless of accuracy (Kaplowitz, 1990). Conflict arises when the other is perceived to have goals or intentions incompatible with one's own goals or payoffs (Pruitt and Rubin, 1986; Winter, 1987), to violate norms of equity or fairness (Aram and Salipante, 1981; Wall and Nolan, 1987), or to harbor harmful intent (Wall and Callister, 1995). In contrast, behavioral factors speak to actual harmful effects: Blocking of one's goals, or control attempts by the other (Alter, 1990); the threat or actual loss of power caused by the other (Blalock, 1989; Fagenson and Cooper, 1987). As a conflict antecedent, communication is mediated by factors such as frequency, content, context, and facility of the communicators. Conflict results when low communication leads to ineffective coordination (Pondy, 1967); however, high communication can produce misunderstanding, especially in a cross-cultural context (Putnam and Poole 1987; Thomas and Pondy, 1977). Structural factors refer to "contextual" characteristics of the social environment (e.g., laws, norms, customs, contracts, requirements, etc.) that constrain or enable interaction and help give it form and content. Previous interactions between parties can work with other interpersonal factors in destructive ways. For example, repeated resolution failures can lead to negative stereotyping, prejudice, and self-fulfilling prophecies that engender new conflicts (Sherif, Harvey, White, Hood, and Sherif, 1961; Smith and Simmons, 1983).

Software Development Research Related Literature

Rather than explore conflict antecedents, research into the role of conflict in team-based software development can be somewhat utilitarian: Often it focuses on the impact of conflict, conflict management, or associated factors on project outcomes, begging the question of how conflict arose to begin with. Some studies do consider conflict precursors, including (1) individual and interpersonal factors, (2) organizational, structural, and contextual factors, and (3) communication.

Individual and Interpersonal Factors

Sawyer (2001, p. 159) states "individual characteristics" are "typically included in most models of conflict among software developers." This might appear inconsistent with general conflict research findings, but a closer examination reveals that very often what Sawyer and others (Barki and Hartwick, 1994; Robey, Farrow, and Franz, 1989; Robey, Smith, and Vijayasarathy, 1993) include in their models are not in fact individually held values, goals, commitments, emotions, personality scores or similar characteristics, but what would clearly be classified as "interpersonal factors" using the Wall and Callister (1995) system: Levels of participation and influence, disagreements between team members, beliefs about other team members or the team in general, and so on. A possible exception is user-developer value divergence as a factor in software evaluation (Wong, 2005).

Organizational, Structural, and Contextual Factors

Organizational, structural, and contextual factors are generally neglected in IS research though some studies note them in passing, e.g., distinguishing between users and developers as members of distinct organizational cultures (Barki and Hartwick, 2001; Gingras and McLean, 1983; Robey et al., 1989), remarking on the contextual nature of conflict and communication issues and the role of scarce resources, organizational rules, and procedures (Yeh and Tsai, 2001), or pointing out that systematic conflict due to divergent, synthesis-resistant goals is endemic to organizations and can result in distrust and conflict among team members (Sawyer, 2001). Process factors such as requirements volatility receive attention as well (Curtis, Krasner, and Iscoe, 1988; Waltz, Elam, and Curtis, 1993).

Communication

Communication has been identified as a critical success factor for software development (Wong and Tein, 2004) with conflict resulting from miscommunication and misunderstanding between stakeholders, particularly users and developers. Such problems are minimized when developers work closely and consistently with users so that minor points can easily be clarified throughout the development process, while relationship building and increasing levels of trust defuse emotional negativity and conflict escalation (Lamp, Altmann, and Hetherington, 2003). However, as noted in the general literature, communication is a double edged sword – when mishandled it can promote, rather than prevent or resolve conflict.

Developer–Tester Interpersonal Conflict

Sawyer (2001) notes that while much IS and software development-specific conflict research has focused on interaction between user and IS staff, the relationship is transformed when software development is carried out by specialized firms so that it becomes just as important to understand conflict among IS staff themselves. This study focuses on the interpersonal conflict between developers and testers in software development.

The software testing process is inherently adversarial, setting the stage for inevitable developer-tester conflict. The role of the tester is to surface flaws and errors in developers' code (Cohen et al., 2004; Parkin, 2001). It is a type of systemic conflict that develops as a result of lateral working relationships (Pondy, 1967). Tensions may arise between developers and testers because "they often have different and even divergent goals that are difficult to synthesize" (Sawyer, 2001, p. 160).

Software developers and testers are very different from each other in terms of mindsets, goals, experiences, and perceptions of their relative importance (Cohen et al., 2004; Pettichord, 2000; Rothman, 2004). First, developers and testers often have different mindsets, as developers always think in terms of "building" something, whereas testers devise means of "breaking" what developers build. Second, developers and testers typically have distinct goals due to the difference in their job functionality. Developers usually seek to maximize "efficiency," that is, to get the work done with the least amount of effort; for example, building the same functionality with the least number of lines of code or in the shortest amount of time. In contrast, testers usually seek to maximize "effectiveness," that is, to deliver the end product with the best quality. Third, developers and testers may differ in work experience; for instance, it is not uncommon that more seasoned developers work side-by-side with less experienced testers. And finally, developers and testers may have different perceptions of their relative importance in the organization; for example, testers often feel that they have to constantly work to gain the same level of respect as that of developers (Cohen et al., 2004).

Individual characteristics testers identified as differentiating them from developers align with Wong's (2005) findings on developer-user value divergence. Testers described themselves as compulsive and detailed while characterizing developers as creative, temperamental, and apt to personalize their code, reacting to error detection as if to personal criticism. These tendencies find expression in work orientation: Testers focus on compliance with user requirements while developers look for ways to exploit the technical possibilities, sometimes violating specifications in the process.

Pettichord (2000) provides a unique way to show that testers and developers are different by comparing a list of twelve traits that make good testers and developers. To summarize the traits, good testers are expected to have broad knowledge of many domains, learn new things quickly, focus on user needs and behavior, think empirically, and concentrate on reporting problems. In contrast, good developers are expected to have specialized knowledge of product internals, gain understanding of new things slowly but thoroughly, focus on system design, think theoretically, and concentrate on understanding problems.

By analyzing in-depth field interviews with ten software testing professionals, Cohen et al. (2004) categorized the sources of conflict between developers and testers into three conflict layers comprising the software testing process, people, and organization. Some of the conflict sources are: (1) Developer and tester competition for the scarce resource of project time, (2) Testers focusing more on user requirements whereas developers focusing more on technical requirements, (3) Differences in tester vs. developer "mental process and personality attributes" related to the process of software development, and (4) The role of differential power and politics, e.g., for some testers, "the struggle for recognition becomes part of the job itself" (Cohen et al., 2004, p. 79).

These findings are both interesting and informative. However, the study has three limitations: (1) The interviewees were all testers, which makes it very likely that only one side of the story was covered, (2) The research lacks a theoretical basis, which makes it difficult to advance knowledge in the conflict domain, and (3) The findings are not backed up with quantitative data, which makes it hard to assess their reliability and validity. These limitations suggest that additional studies are needed. Accordingly, we posed the following research question for an exploratory study to build upon Cohen et al. (2004): *What are the sources of interpersonal conflict between developers and testers in software development?* Although we are not able to address all the limitations mentioned above in this preliminary study, we plan to do so in future studies.

RESEARCH METHODOLOGY

The purpose of this exploratory study is to build upon the work of Cohen et al. (2004) to construct a conceptual framework for subsequent empirical research. Accordingly, we used a qualitative approach to address our research question, utilizing content analysis techniques consistent with guidelines established by Krippendorff (1980) and Neuendorf (2002).

By analyzing developer-tester conflict scenarios collected from both code developers and testing professionals, a key limitation of the earlier study is addressed.

Samples and Data Collection

Data was collected from a convenience sample of participants enrolled in a week-long system testing training course developed for employees of a large, globally branded company. The sample is a good fit for this study for the following reasons:

1. All research subjects are employed by a single organization subdivided into functionally specialized units including software development, and QA (quality assurance). Automated tools are routinely used for both detailed application design and software testing.
2. As participants in training designed specifically to teach testing skills, all research subjects are likely to be actively engaged in either software development, QA (software testing), or both.
3. Most course participants were either code developers or software testing professionals; the remainder work directly with development teams as either business analysts or IS managers.

Written conflict scenarios were collected in the following manner: Immediately after completion of the "conflict and conflict management" training module, the course instructor asked each participant to provide a written description of a software development conflict experience, including information about the issues involved and reactions of both parties.

Data Analysis and Categorization

First the original hand-written conflict scenarios were transcribed into an MS Word file. Soft copies of the transcript were then distributed to two persons meeting the cognitive and educational criteria for review and categorization of scenario content: (1) A doctoral student very familiar with the information systems development interpersonal conflict literature, and (2) A developer with a master's degree but no past experience interacting with testers and little knowledge of the interpersonal conflict literature. These two independently analyzed the transcripts, categorizing each according to a schema based on Cohen et al. (2004): Software testing process, people, and organization.

To ensure intercoder reliability, the categorized scenarios were then reviewed and validated by a senior MIS faculty member actively involved in the system testing training program. The results presented below were synthesized from these three input sources. Divergence of opinion regarding categorization of a conflict scenario was resolved by majority rule.

RESULTS

Sixty-four conflict scenarios were collected. Of these, fourteen were discarded after initial analysis due to irrelevancy; i.e., they are not related to developer-tester conflict. The remaining fifty scenarios were sorted according to reported sources of developer-tester conflict. Although initially guided by the "software testing process, people, organization" schema described by Cohen et al. (2004), scenario analysis resulted in development of an alternative three-category conceptual structure that provides a better fit for the data: Process, people, and communication.

Figure 1 depicts the relationship between the three as a layered subset structure in which each constrains or influences the category or categories contained within it: (1) "Process" provides an organizational context for software development and is therefore depicted as the outermost layer; (2) "People" is contained within process because when enforced, process constrains individual and group behavior; and (3) "Communication" is the third layer because it is a key component of human behavior and thus a function of, and contained within, the people layer. This structure is intuitive and experientially supported. Working from the bottom up, project team communication is performed by people, whose choices of when, how, and what to communicate are constrained and guided by established process. Reversing direction and starting with the top layer, process directly influences team behavior (people) and thus indirectly influences the nature, frequency, content, direction, and effectiveness of communication.

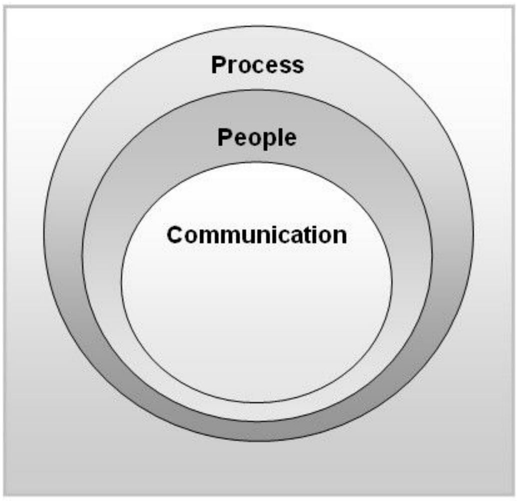

Figure 1. Three-Layer Conflict Model

The three layers or categories were then further divided into nine subcategories based on recurring themes or keywords (see Table 1 for detail). Following is a presentation of findings in a category-subcategory-example format.

Process (documentation). Conflict arises when testers fail to provide adequate documentation so that developers can understand what is wrong, and how to reproduce the error.

Process (procedure). Some of the reported developer-tester conflicts are related to procedure. They include: (1) Tester opened a change request (CR) without thorough consideration, (2) Developer provided an incorrect version of code to testers for testing, (3) Developers and testers don't know the process or don't follow it correctly when a CR is opened, (4) Developers fix defects without properly documenting the code changes, (5) Developers and testers fail to involve third parties whose participation is needed for conflict resolution, (6) Testers ask developers to change requirements after code completion, and (7) Developers access the test environment without invitation or clearance from the test organization.

Process (resource). Cohen et al. (2004) noted that developer-tester conflict arises when developers and testers compete for scare resources; two commonly reported factors in the general conflict literature are budget and time. In accordance with this observation, one scenario reports conflict that resulted from failure to include testing in the project budget. Another notes that last minute requirements changes and resulting delays in development squeezed the testing schedule.

Process (standards). Finally, lack of code standards or divergence in standards expectations may cause conflict between developers and testers. For instance, testers may use testing methods that make assumptions about the code while developers code according to the detailed design, not taking code standards or testing expectations into account.

People (attitude). Developers and testers have strong negative attitudes toward each other. One scenario describes a developer who holds such a low opinion of testers that he will not offer any suggestions, leaving them to work without the benefit of his support and expertise. In another scenario, a tester is said to derive negative enjoyment from finding errors in developers' code. A third case describes a heated argument during a code review, in which a tester stated to the developer that "your code" has such and such defects. A fourth scenario states that whenever anything goes wrong, developers and testers point fingers at each other; testers blame a logic error in the code, while developers say it is a data issue. And finally, developers may constantly question the validity of the testing organization's metrics.

People (view). Developers and testers can have conflicting views about appropriate levels of validation. At one point during a development status meeting, a frustrated development project lead almost screamed that testers were testing "too many combinations." Developers and testers may also have different opinions about what defects should be fixed first, and how the fixes should be validated.

People (focus). Developers and testers bring differential focus to software quality assessment. Developers often focus on technical requirements; while as end user representatives, testers often focus on business requirements.

Category	Subcategory	Examples (number refers to original scenario)
Process	Documentation	Testers fail to provide documents to show developers what's wrong, and how to reproduce the errors. (39)
	Practice	Testers open change requests (CRs) without thorough consideration. (30)
		Developers provided incorrect version of code to testers. (46)
		Developers and testers don't know the process or don't follow it correctly when a CR is opened. (9)
		Developers fixed some defects without properly documenting what they did. (15)
		Failed to involve third party responsible for the problem. (27)
		Tester asks developer to change requirements after code is complete. (58)
	Resource	No budget for testing. (1)
		Testing time was squeezed because of last minute requirements changes, causing development to drag on. (28)
	Standards	Lack of standards and code expectations. (17)
People	Attitude	Testers have fun at developer expense by finding errors in their code. (18)
		Developers question testing organization's metric for testing. (33)
		Code personalized: During a heated argument with a tester, a developer is told that "your code" has such and such defects. (49)
		Developer who has a low opinion of testers. (57)
		What's wrong? Testers say it is a logic error in the code; developers say it's a data issue. (6)
	View	Conflict views on the level of testing: During a development status meeting, the development project lead almost screams that testers are testing too many combinations. (19)
		Different point of views on what defects should be fixed first, and how the fixes should be validated. (44)
	Focus	Different focus: Developers on technical requirements, testers on business requirements. (23)
	Knowledge	Testers lack knowledge of a particular application. (3)
		Testers lack understanding of the business requirements. (8)
Communication	Communication	Test lead left out of the communication loop. (10)

		Not knowing each other's process or responsibilities. (14)
		Last minute communication. (16)
		Changes of procedure not communicated. (20)
		Late communication of new requirements. (35)
		Prompt feedback not provided. (47)
		Different languages spoken. (50)
		Developers changed code without communicating to testers. (62)

Table 1. Developer-Tester Conflict Sources in Software Development

People (knowledge). Developers may accuse testers of lack of knowledge about the application, or lack of understanding of the business requirements.

Communication (communication). Many communication related conflicts between developers and testers were reported, including: (1) Test lead was left out of the communication loop, (2) Not knowing each other's process or responsibilities, (3) Communicating only at the last minute, (4) Not communicating with each other about procedural changes, (5) Delay in communication of new requirements, (6) Failure to provide timely feedback, (7) Speaking different languages which leads to misunderstanding or not being able to understand each other, and (8) Developers making code changes without notifying testers.

DISCUSSION

The purpose of this study was to identify sources of interpersonal conflict between software developers and testers. Results indicate that developer-tester conflict antecedents fall into three major categories: Process, people, and communication. In particular, developers and testers experience conflict primarily because: (1) They don't have standardized procedures to follow, (2) they have distinctive attitudes, views, foci, and knowledge bases, and (3) they don't communicate effectively or efficiently.

Comparison of Two Models

Our research focus and data analysis were significantly influenced by Cohen et al. (2004) so it is instructive to compare methodology and results between the two studies. In both a qualitative approach utilized open-ended questions to gather information about conflict experiences from professionals involved in software development projects. While Cohen et al. conducted in-depth field interviews with ten software testing professionals, we collected fifty written conflict scenarios from both testers and developers. Based on their analysis, Cohen et al. (2004) constructed the three-layer conflict model depicted in Figure 2 below. Similar to our process layer (Figure 1), "organization" appears to function as a contextual base for the remaining layers; but in contrast to our model "software testing process" is contained within "people" and communication is not featured as a major conflict source.

We believe that both our data (in terms of sample size and characteristics) and more particularly, our model represent a significant advance from the departure point provided by Cohen et al. (2004). Although small sample size is not a serious flaw for exploratory studies, our fifty written scenarios from both developers and testers employed by an industry leading company compare favorably with the earlier study's interviews with ten software testers only. Regarding model construction, Cohen et al. (2004) do not provide a rationale for the apparent ranking of their three conflict layers (Figure 2) or the nature of the relationships between them, nor are the conceptualizations informing the arrangement readily inferred. While "organization" (layer 1) appears to function legitimately both as a major conflict source and as an overall context within which team conflict arises, positioning of the other two layers is problematic because experientially, it is difficult to support the assertion that "software testing process" (layer 3) is primarily a function or subset of "people" (layer 2). Rather, people act (or fail to act) in compliance with process, a key organizational feature. Nor is it inevitable that formal process will be significantly altered if the people assigned to a project are replaced; the usual expectation is that process will remain relatively stable regardless of team composition. This suggests that the ranking should be altered so that "software testing process" functions as the second layer, contained by "organization" (layer 1) while constraining the behavior of "people" (layer 3).

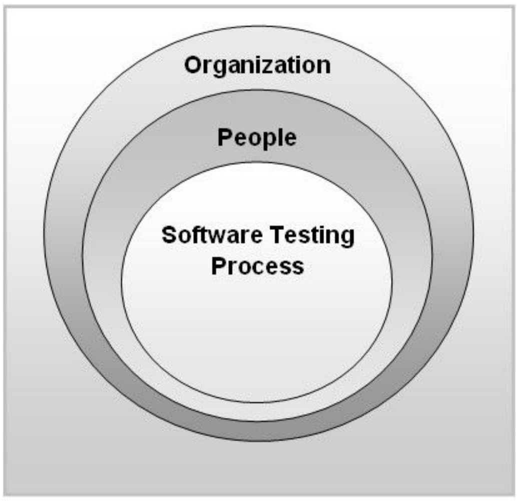

Figure 2. Layers of Conflict (Cohen et al., 2004)

Playing devil's advocate for a moment, it's true that in their capacity as structural agents (Giddens, 1986) project team members can and often do function as owners of (rather than just clients constrained by) formal or informal processes existing at a variety of levels: Organizational, departmental, project, subteam, etc. It is also true that when established process does not adequately serve the needs of a particular project, or when incompatible processes interfere with team effectiveness, team members may choose to solve the problem with formal or informal process innovations. Just the same, vis-à-vis Cohen et al. (2004) our layer ranking recommendations stand. By its nature, process constrains individual behavior so as to produce predictable results; that is the entire purpose of creating, documenting, and enforcing it.

A final contrast between this study and Cohen et al. (2004) is our finding (consistent with the general conflict literature) that communication functions as a major source of conflict between developers and testers. It should be noted, however, that all but one of the communication-related scenarios we describe in the Results section above appear to be symptoms of an underlying cause: Lack of unified, appropriately designed and effectively applied project structure and process. Thus, through second layer mediation ("people," Figure 1), process gaps or flaws can negatively impact project communication, arguably rendering suspect the decision to classify such issues as arising from layer 3 (communication) rather than layer 1 (process). Our classification decision was guided by respondent perception as indicated by use of the keyword "communication" within the descriptive text.

Implications

As with the earlier qualitative treatment by Cohen et al. (2004), this study aligns well with Lyytinen's (1999) advocacy for research that is clearly linked to practice, both in its methodological emphasis on practitioner-informed antecedent categories and the construction of an experientially-based conflict model .

The findings of this study have important implications for research. Conflict appears to originate from the organizational layers of process, people, and communication, which can be perfectly mapped to interpersonal factors identified by previous conflict studies including structure, perceptual interface, and communication, respectively. Our findings partially support those of Cohen et al. (2004) in terms of people and process as conflict sources. However, this study suggests that communication is an additional major source of interpersonal conflict. This supports prior studies identifying communication is an antecedent of interpersonal conflict (Baron, 1989; Pondy, 1967; Putnam and Poole, 1987). Therefore, it is critical that development managers and technical leads create an environment which facilitates effective communication between developers and testers and at the same time, decreases the chance of misunderstanding between them.

Our findings also have important implications for practice. Accurate identification of potential conflict sources enables development managers and technical leads to proactively target management strategies, preventing dysfunctional conflict from developing, or intervening before it can escalate and damage project outcomes, software quality, and ongoing working relationships. Despite differences in methodology and conflict layer modeling, commonalities between our results and those reported by Cohen et al. (2004) suggest that developer-tester conflict is rooted in both context (process or organizational characteristics) as well as human interaction (behavioral, attitudinal, or communication characteristics). From a practical standpoint, organizations would be wise to take these causal categories into account in creating both long-term strategies and short-term tactics to prevent, mediate, or resolve software developer-tester conflict.

Lyytinen (1987) noted the change-inhibiting effects of prevailing IS social arrangements, suggesting improvements in administrative models and IS processes to enable more flexible interaction and negotiation. With reference to the conflict antecedents described above, this could include procedural innovations to address flaws such as poor process fit or inconsistent process compliance, poorly defined team roles, lack of unified cross-functional authority structures, and the like. Process tools can also target communication with clearly delineated responsibility for reporting, statusing, decision documenting, and information distribution, while clarity and unity of vision is enabled by adopting standards and technology to achieve on-demand information access. In addition, training, mentoring, and formal policy can support more effective people management and negotiation behaviors.

Limitations

The study has several limitations. First, it is based on a convenience sample, limiting the generalizability of our findings. Second, the sample size is relatively small (fifty useable conflict scenarios), making it difficult to quantify the validity of the findings; however, this is not a serious concern with an exploratory study which does not attempt theory testing.. Third, it didn’t differentiate conflict sources specified by developers from those described by testers, which would have enabled response comparisons. And finally, it didn’t collect standard control data (e.g., age, gender, tenure, education, etc.) that would support identification of sub-constituencies within the developer and tester groups, or of demographic differentials that could interact with other conflict factors in the software testing context. All of these limitations will be addressed in subsequent studies.

Further Studies

Further studies are planned that will take a more systematic approach to collecting conflict scenarios, with a combined open-ended / structured online questionnaire administered via surveymonkey.com. We will distribute the survey link to potential respondents representing two key stakeholders in software development, i.e., developers and testers. Our targets are a minimum of 200 developers and 200 testers. Each respondent will be asked to describe a personal developer-tester conflict experience. Questions will include: (1) What was the issue that caused the problem? (2) Who was involved in the conflict? (3) How did you react; how did the other party react? (4) Was the conflict escalated to management level? (4) Was the conflict resolved; if so, how and who was involved in the resolution? (5) How often does this type of conflict occur? We will also collect demographic data for the purpose of providing descriptive statistics for the sample and testing for potential control variables. When data collection is complete, we will analyze the survey responses to identify and categorize sources of conflict, comparing the perceptions of developers and testers in a structured, easy-to-understand, and easy-to-communicate way.

CONCLUSION

Interpersonal conflict between developers and testers is an ubiquitous phenomenon in software development. It can develop from factors associated with three major contextual layers – process, people, and communication. Although complete elimination of developer-tester conflict is impossible, overall software development effectiveness and efficiency can be greatly improved if conflict is managed at the source. To fulfill this goal, the first step is to clearly identify and thoroughly understand the sources of conflict. Our preliminary results have demonstrated the utility of this approach.

REFERENCES

1. Alter, C. (1990) An exploratory study of conflict and coordination interorganizational service delivery systems, *The Academy of Management Journal*, 33, 3, 478-502.
2. Aram, J.D. and Salipante, P.F. (1981) An evaluation of organizational due process in the resolution of employee/employer conflict, *The Academy of Management Review*, 6, 2, 197-204.
3. Augsburger, D.W. (1992) *Conflict mediation across cultures: Pathways and patterns*, Westminster/John Knox Press, Louisville, Kentucky.
4. Barki, H. and Hartwick, J. (1994) User participation, conflict, and conflict resolution: The mediating roles of influence, *Information Systems Research*, 5, 4, 422-438.

5. Barki, H. and Hartwick, J. (2001) Interpersonal conflict and its management in information system development, *MIS Quarterly*, 25, 2, 195-228.
6. Baron, R.A. (1989) Personality and organizational conflict: Effects of the type A behavior pattern and self-monitoring, *Organizational Behavior and Human Decision Processes*, 44, 2, 281-296.
7. Beath, C.M. and Orlikowski, W.J. (1994) The contradictory structure of systems development methodologies: Deconstructing the IS-user relationship in information engineering, *Information Systems Research*, 5, 4, 350-370.
8. Blalock, H.M. Jr. (1989) *Power and conflict: Toward a general theory*, Sage, Thousand Oaks, California.
9. Cohen, C.F., Birkin, S.J., Garfield, M.J., and Webb, H.W. (2004) Management conflict in software testing, *Communications of the ACM*, 47, 1, 76-81.
10. Collins, F. (1986) Human issues play lead role in information systems melodrama, *Data Management*, 24, 8, 26-29.
11. Coombs, C.H. and Avrunin, G.S. (1988) *The structure of conflict*, Lawrence Erlbaum Associates, Hillsdale, New Jersey.
12. Curtis, W., Krasner, H. and Iscoe, N. (1988) A field study of the software design process for large systems, *Communications of the ACM*, 31, 11, 1268-1287.
13. De Dreu, C.K.W. and Weingart, L.R. (2003) Task versus relationship conflict, team performance and team member satisfaction: A meta-analysis, *Journal of Applied Psychology*, 88, 4, 741-749.
14. DeLone, W.H. and McLean, E.R. (1992) Information systems success: The quest for dependent variable, *Information Systems Research*, 3, 1, 60-95.
15. DeLone, W.H. and McLean, E.R. (2003) The DeLone and McLean model of information systems success: A ten-year update, *Journal of Management Information Systems*, 19, 4, 9-30.
16. Derr, C.B. (1978) Managing organizational conflict: Collaboration, bargaining, and power approaches, *California Management Review*, 21, 2, 76-83.
17. Deutsch, M. (1990) Sixty years of conflict, *International Journal of Conflict Management*, 1, 3, 237-263.
18. Dos Santos, B.L. and Hawk, S.R. (1988) Differences in analysts' attitude towards information systems development: Evidence and implications, *Information & Management*, 14, 1, 31-41.
19. Fagenson, E.A. and Cooper, J. (1987) When push comes to power: A test of power restoration theory's explanation for aggressive conflict escalation, *Basic and Applied Social Psychology*, 8, 4, 273-293.
20. Giddens, A. (1986) *The constitution of society: Outline of the theory of structuration*, University of California Press, Berkeley, California.
21. Gingras, L. and McLean, E. (1982) Designers and users of information systems: A study in differing profiles, in M. Ginzberg and C.A. Ross (Eds.) *Proceedings of the Third International Conference on Information Systems*, Ann Arbor, Michigan, USA, 169-181.
22. Hulin, C.L. (1990) Adaptation, persistence, and commitment in organizations, in Dunnette, M.D. and Hough, L.M. (Eds.) *Handbook of Industrial and Organizational Psychology* (2nd ed., Vol.2), Consulting Psychologies Press, Palo Alto, California, 445-506.
23. Hwang, M.I. and Thorn, R.G. (1999) The effect of user engagement on system success: A meta-analytical integration of research findings, *Information & Management*, 35, 4, 229-236.
24. Ives, B. and Olson, M.H. (1984) User involvement and MIS success: A review of research, *Management Science*, 30, 5, 586-603.
25. Jehn, K. (1995) A multimethod examination of the benefits and detriments of intragroup conflict, *Administrative Science Quarterly*, 40, 2, 256-282.
26. Jehn, K.A., Rupert, J., and Nauta, A. (2006) The effects of conflict asymmetry on mediation outcomes: Satisfaction, work motivation and absenteeism, *International Journal of Conflict Management*, 17, 2, 96-109.
27. Kaplowitz, N. (1990) National self-images, perception of enemies, and conflict strategies: Psychopolitical dimensions of international relations, *Political Psychology*, 11, 1, 39-82.
28. Krippendorff, K. (1980) *Content analysis: An introduction to its methodology*, Sage, Beverly Hills, California.
29. Lamp, J., Altmann, G., and Hetherington, T. (2003) Functional group conflict in information systems development, in P. Love, C. Standing, and N. Lethbridge (Eds.) *Proceedings of 14th Australasian Conference on Information Systems*, Perth, Western Australia, 1-8.

30. Luftman, J. (2008) Yes, the tech skills shortage is real, *InformationWeek*, January 12, 2008.
31. Lyytinen, K. (1987) Different perspectives on information systems: Problems and solutions, *ACM Computing Surveys*, 19, 1, 5-46.
32. Lyytinen, K. (1999) Empirical research in information systems: On the relevance of practice in thinking of IS research, *MIS Quarterly*, 23, 1, 25-27.
33. Maslach, C., Schaufeli, W.B., and Leiter, M.P. (2001) Job burnout, *Annual Review of Psychology*, 52, 1, 397-422.
34. Neuendorf, K.A. (2002) *The content analysis guidebook*, Sage, Thousand Oaks, California.
35. Parkin, R. (2001) Developers and testers – Who should do what? Iv & V Australia Pty Ltd., 1-3.
36. Pettichord, B. (2000) Testers and developers think differently: Understanding and utilizing the diverse traits of key players on your team, *Software Testing & Quality Engineering*, 2, 1, 42-45.
37. Pondy, L.R. (1967) Organizational conflict: Concepts and models, *Administrative Science Quarterly*, 12, 2, 296-320.
38. Pruitt, D.G. and Rubin, J.Z. (1986) *Social conflict: Escalation, stalemate, and settlement*, McGraw-Hill, New York, New York.
39. Putnam, L.L. and Poole, M.S. (1987) Conflict and negotiation, in Jablin, F.M., Putnam, L.L., Roberts, K.H., and Porter, L.W. (Eds.) *Handbook of Organizational Communication: An Interdisciplinary Perspective*, Sage, Newbury Park, California, 549-599.
40. Rahim, M.A. and Bonoma, T.V. (1979) Managing organizational conflict: A model for diagnosis and intervention, *Psychological Reports*, 44, 3, 1323-1344.
41. Robey, D. and Farrow, D. (1982) User involvement in information systems development: A conflict model and empirical test, *Management Science*, 28, 1, 73-85.
42. Robey, D., Farrow, D., and Franz, C.R. (1989) Group process and conflict in system development, *Management Science*, 35, 10, 1172-1191.
43. Robey, D., Smith, L.A., and Vijayasarathy, L.R. (1993) Perceptions of conflict and success in information systems development projects, *Journal of Management Information Systems*, 10, 1, 123-139.
44. Robey, D., Welke, R., and Turk, D. (2001) Traditional, iterative, and component-based development: A social analysis of software development paradigms, *Information Technology and Management*, 2, 1, 53-70.
45. Rothman, J. (2004) No more second-class testers! *Better Software*, January 2004.
46. Sawyer, S. (2001) Effects of intra-group conflict on packaged software development team performance, *Information Systems Journal*, 11, 2, 155-178.
47. Sherif, M., Harvey, O.J., White, B.J., Hood, W.R. and Sherif, C.W. (1961) *Intergroup conflict and cooperation: The robbers cave experiment*, Institute of Group Relations, Norman Oklahoma.
48. Simmel, G. (1964) *Conflict and the web of group affiliations*, The Free Press, New York, New York.
49. Smith, H.A. and McKeen, J.D. (1992) Computerization and management: A study of conflict and change, *Information & Management*, 22, 1, 53-64.
50. Smith, K.K. and Simmons, V.M. (1983) A rumpelstiltskin organization: Metaphors on metaphors in field research, *Administrative Science Quarterly*, 28, 3, 377-392.
51. Thomas, K.W. and Pondy, L.R. (1977) Toward an "intent" model of conflict management among principal parties, *Human Relations*, 30, 12, 1089-1102.
52. Wall, J.A. and Callister, R.R. (1995) Conflict and its management, *Journal of management*, 21, 3, 515-558.
53. Wall, V.D. and Nolan, L.L. (1987) Small group conflict: A look at equity, satisfaction, and styles of conflict management, *Small Group Behavior*, 18, 2, 188-211.
54. Walz, D., Elam, J., and Curtis, B. (1993) The dual role of conflict in group software requirements and design activities, *Communications of the ACM*, 36, 10, 63-76.
55. Wang, E.T.G., Chen, H.H.G., Jiang, J.J., and Klein, G. (2005) Interaction quality between IS professionals and users: Impacting conflict and project performance, *Journal of Information Science*, 31, 4, 273-282.
56. Winter, D. (1987) Enhancement of an enemy's power motivation as a dynamic of conflict escalation, *Journal of Personality and Social Psychology*, 52, 1, 41-46.

57. Wong, B. (2005) Understanding stakeholder values as a means of dealing with stakeholder conflicts, *Software Quality Journal*, 13, 4, 429-445.

58. Wong, B. and Tein, D. (2004) Critical success factors for enterprise resource planning projects, *Journal of the Australian Institute of Project Management*, 24, 1, 28-31.

59. Wong, C.L., Tjosvold, D., and Lee, F. (1992) Managing conflict in a diverse work force: A Chinese perspective in North America, *Small Group Research*, 23, 3, 302-321.

60. Yeh, Q.-J. and Tsai, C.-L. (2001) Two conflict potential during IS development, *Information & Management*, 19, 2, 135-149.