Interpersonal Conflict between Developers and Testers in Software Development

Xihui Zhang, Jasbir S. Dhaliwal, Mark L. Gillenson
Department of Management Information Systems
Fogelman College of Business and Economics
University of Memphis, Memphis, TN 38152
{xihui.zhang, jdhaliwI, mark.gillenson}@memphis.edu

Abstract
Software development is a complex process that necessitates interaction between diverse individuals playing differentiated roles. These include users, business analysts, system analysts, designers, programmers, and testers. The trend in software development methodologies is towards those that require more frequent and sophisticated interaction between developers (a category that includes system analysts, designers and programmers) and testers. Given the inherent differences between developers and testers and their differentiated roles, the increased interaction is likely to lead to a greater level of interpersonal conflict between them. This research explores the theoretical and pragmatic nature of such developer-tester conflict in terms of both task and relationship elements as well as its impact on software development outcomes. The effectiveness of various conflict management styles that can help regulate the level of developer-tester conflict is also explored. Based on an extensive literature review, this paper advances a theory-driven research model for conflict in developer-tester relations and presents a set of hypotheses that are to serve as the basis for empirical studies of this critical phenomenon.

1. Introduction
Interpersonal conflict between developers and testers is inevitable and pervasive in software development given task and individual differences. A lack of understanding about this conflict and its dysfunctional impact can lead to organizational ineffectiveness and inefficiencies. The objective of this research is to facilitate understanding of this phenomenon by undertaking a literature review of interpersonal conflict and conflict management styles in relation to practices of software development and testing.

Traditionally, testing has been viewed as a separate and distinct stage at the tail-end of the software development life cycle. The latest trends in software development, however, suggest that testing personnel and activities have to be engaged earlier in the development process [21], [30], [33]. A key argument for doing this is to catch defects early in software development because fixing a defect detected in the later phases tends to be 10 to 100 times more costly [31]. New methodologies based on prototyping such as RAD, JAD, Xtreme method, and agile development also give testing a more significant role earlier in the development process. This trend suggests that a greater level of interaction between developers and testers is to be expected and this will obviously lead to a greater level of interpersonal conflict which needs to be managed.

Another key reason why it is important to understand interpersonal conflict between developers and testers is because this conflict can have negative consequences not only in relation to the end product but also in relation to the job satisfaction of both developers and testers. Rahim and Bonoma [24] argue that the relation between conflict and organizational effectiveness approximates an inverted-U function, with too much or too little conflict negatively impacting organizational effectiveness. Therefore, proactive management for the purposes of maintaining an appropriate level of conflict is crucial for optimizing organizational effectiveness. However, it is imperative that we understand the nature of this conflict before we can appropriately design managerial tactics for measuring and managing it.

The following sections include justification for research, theoretical foundations, research model and hypotheses, and research methodology.
2. Justification for research

In most organizations, human resources are regarded as one of the most valuable resources for undertaking projects. As assigned projects, usually under pressing deadlines, are getting more complex, people have to collaborate with each other because a single individual is incapable of delivering projects on time, under budget, and with the required functionality and features. When people interact, however, interpersonal conflict arises. Interpersonal conflict is the result when interdependent parties have different goals, mindsets, values, preferences, backgrounds, and experiences [2], [7].

Software testing is indispensable to ensuring software quality [7]. In the process of software development, developers and testers interact constantly. However, developers and testers are very different from each other in terms of goals, experiences, and perceptions of their relative importance [7], [20], [27]. Because of the difference in their job functionality, developers and testers typically have distinctive goals. A developer is usually seeking to maximize “efficiency,” that is, to get the work done with the least effort, for example, building the same functionality with the least number of lines of code. In contrast, a tester is usually seeking to maximize “effectiveness,” that is, to deliver the end product with the best quality. In addition, developers and testers may differ in 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 [7]. Therefore, interpersonal conflict between developers and testers represents the most prevailing conflict in software development.

By analyzing in-depth field interviews with ten software testing professionals, Cohen et al. [7] categorized the sources of conflict between developers and testers into three conflict layers, including the software testing process, people, and organization. Two sources of conflict in the process layer are: 1) developers and testers compete for the scarce resource of project time, and 2) testers focus more on user requirements whereas developers focus more on technical requirements. Two sources of conflict in the people layer are: 1) testers and developers have different “mental process and personality attributes” over the process of software development, and 2) many developers become too protective when testers find defects in their code. In the organization layer, power and politics play a role, and managers matter. This research is very informative, but it has three limitations: 1) the interviewees in the study were all testers, which makes it highly possible 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 based on quantitative data.

Pettichord [20] provides yet another way to understand that testers and developers are different by comparing a list of twelve traits that make good testers and developers, as shown in Table 1. To summarize, good testers are supposed to have a broad knowledge of many domains, learn new things fast, focus on user needs and behavior, think empirically, and concentrate on reporting problems. In contrast, good developers are supposed to have a specialized knowledge of product internals, gain thorough understanding of new things slowly but surely, focus on system design, think theoretically, and concentrate on understanding problems.

<table>
<thead>
<tr>
<th>Good Testers</th>
<th>Good Developers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain knowledge</td>
<td>Knowledge of product internals</td>
</tr>
<tr>
<td>Get up to speed quickly</td>
<td>Thorough understanding</td>
</tr>
<tr>
<td>Ignorance is important</td>
<td>Expertise is important</td>
</tr>
<tr>
<td>Model user behavior</td>
<td>Model system design</td>
</tr>
<tr>
<td>Focus on what can go wrong</td>
<td>Focus on how it can work</td>
</tr>
<tr>
<td>Focus on severity of problem</td>
<td>Focus on interest in problem</td>
</tr>
<tr>
<td>Empirical</td>
<td>Theoretical</td>
</tr>
<tr>
<td>What’s observed</td>
<td>How it’s designed</td>
</tr>
<tr>
<td>Skeptics</td>
<td>Believers</td>
</tr>
<tr>
<td>Tolerate tedium</td>
<td>Automate tedium</td>
</tr>
<tr>
<td>Comfortable with conflict</td>
<td>Avoid conflict</td>
</tr>
<tr>
<td>Report problems</td>
<td>Understand problems</td>
</tr>
</tbody>
</table>

Software development typically includes five phases: analysis, design, coding, testing, and implementation [16]. Traditionally, testing does not start until the coding phase is complete. Some empirical studies have shown that engaging testers earlier and throughout the software development process is beneficial. Waligora and Coon [33] present quantitative evidence that, by starting testing earlier in the development life cycle, project performance, in
terms of cost and cycle time, is improved, without sacrificing quality. Vijay [32] suggests that we should tap into the testing phase if we want to cut down the cycle time of software development because “the testing activity consumes approximately around 40% of the whole project time,” and “around 60-75% of the activities (like the test-plan, test design) doesn’t require any code.”

Without doubt, engaging testers earlier and throughout the software development process will substantially increase the interaction frequency between developers and testers, which in turn will, not surprisingly, intensify the interpersonal conflict between them.

A debate has been going on for decades on whether conflict in an organization is good or bad in terms of its positive or negative impact on the outcomes. (Outcomes typically include time to market, cost, quality, satisfaction, and performance, etc.) Past empirical studies have yielded mixed findings with most studies showing that conflict has a negative impact on outcomes [1], [2], [14]. Other studies show that conflict has a positive impact on outcomes [13], [18], while some others show that there is no significant relationship between conflict and outcomes [19].

The objectives of this study are: 1) to identify pertinent sources of interpersonal conflict in developer and tester relations, 2) to uncover the nature and form of such conflict, 3) to investigate the impact of such conflict on software development outcomes; and 4) to develop guidelines about appropriate conflict management styles and methods for managing such conflict based on empirical research findings.

The research will make contributions to both theory and practice. In relation to theory development, it will be the first empirical study that tests theoretically-driven hypotheses about interpersonal conflict in developer-tester interactions. It will specifically test the applicability of Barki and Hartwick’s [2], [3] interpersonal conflict theory to the context of developer-tester relations. It will also investigate the nature of the impact of such conflict in relation to software development outcomes. For example, it will validate theoretical notions of what can be termed as the “inverted-U” phenomenon which posits that too low or high levels of conflict have negative impacts while moderate levels are optimal.

In relation to practice, this research will help software development managers and technical leads better appreciate the sources of interpersonal conflict amongst developer and tester groups so that they are in a better situation to be able to prevent, mediate or resolve this conflict. It will also assist them in understanding how such conflict may impact the success of their software development efforts. Finally, the findings will be useful for the design of pragmatic and effective managerial tactics that can be used for managing such conflict.

3. Theoretical foundations

This research has three theoretical foundations. They are interpersonal conflict, conflict management styles, and the practice of software testing. This section explains each of the themes sequentially.

3.1. Interpersonal conflict

Interpersonal conflict has been the subject of research and a concern of practice for more than half a century. There are numerous books, journal articles, conference papers, and even a journal (International Journal of Conflict Management) that are dedicated to it [3]. There are many definitions of interpersonal conflict, each with a different perspective and focus. Barki and Hartwick [3] suggest that there is a need for “a clear conceptualization and operationalization of the construct of interpersonal conflict” because the lack of it “makes it difficult to compare the results of different studies and hinders the accumulation of knowledge.” After an exhaustive literature review on the research of interpersonal conflict that spanned from 1990 to 2003, they came up with a two-dimensional framework and a typology of interpersonal conflict. Our theoretical basis for interpersonal conflict will be based on this framework. The rest of this subsection concentrates on presenting this two-dimensional framework and typology of interpersonal conflict.

Barki and Hartwick [3] present the construct of interpersonal conflict as a two-dimensional framework. The first dimension is of the three fundamental properties of interpersonal conflict: disagreement, interference, and negative emotion. The second dimension is of two main targets of interpersonal conflict: task and interpersonal relationship. A typology for conceptualizing and assessing interpersonal conflict in organizations is shown in Table 2.

<table>
<thead>
<tr>
<th>Task Content or Interpersonal Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpersonal Conflict’s Target</td>
</tr>
</tbody>
</table>
Interpersonal Conflict's Properties

<table>
<thead>
<tr>
<th>Cognition/Disagreement</th>
<th>Task Process</th>
<th>Behavior/Interference</th>
<th>Task Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disagreement with the other about what should be done in a task or how a task should be done</td>
<td>Disagreement with the other’s personal values, views, preferences etc.</td>
<td>Preventing the other from doing what they think should be done in a task or how a task should be done</td>
<td>Preventing the other from doing things unrelated to a task</td>
</tr>
<tr>
<td>Behavior/Interference</td>
<td>Behavior/Interference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preventing the other from doing what they think should be done in a task or how a task should be done</td>
<td>Preventing the other from doing things unrelated to a task</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Affect/Negative Emotion</td>
<td>Affect/Negative Emotion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anger and frustration directed to the other about what should be done in a task or how a task should be done</td>
<td>Anger and frustration directed to the other as a person</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From this two-dimensional framework and the typology of interpersonal conflict, they define interpersonal conflict as “a dynamic process that occurs between interdependent parties as they experience negative emotional reactions to perceived disagreements and interference with the attainment of their goals” [3, p. 234]. They conceptualize interpersonal conflict as a construct that incorporates the simultaneous presence of cognitions, emotions, and behaviors relevant to conflict contexts.

### 3.2. Conflict management styles

There are various styles for managing interpersonal conflict. These conflict management styles are adopted by conflicting individuals when dealing with their interpersonal conflict. These conflict management styles are different from third-party conflict intervention strategies.

Numerous models exist in the literature on conflict management styles [6], [9], [22], [24], [29]. These models can be categorized into one dimensional [9], two dimensional [6], [29], [24], [22], or three dimensional [12]. Deutsch [9] proposed a one-dimensional model and differentiated cooperation and competition as two conflict management styles. Blake and Mouton [6] proposed a two-dimensional model based on “concern for product” and “concern for people” and classified conflict management styles into five types: forcing, withdrawing, smoothing, compromising, and problem solving. Thomas [29] further refined this two-dimensional model by considering intensions (cooperativeness or assertiveness) of the conflicting parties.

However, the most widely used model, which will be adopted for this study, is that proposed by Rahim and Bonoma [24]. It is a two-dimensional model, with the two dimensions being “concern for self” and “concern for others.” The concern for self dimension is about the degree (high or low) in which an individual attempts to satisfy the concerns or needs of her own. The concern for others dimension is about the degree (high or low) in which an individual attempts to satisfy the concerns or needs of others. Crossing these two dimensions yields five different conflict management styles, including integrating, obliging, dominating, avoiding, and compromising, as shown in Figure 1. (See [26], [25], and [2] for detailed descriptions of when each style occurs, what each style involves, and what the outcomes are for each style.)

![Figure 1: A two dimensional model of the conflict management styles (adapted from [24], [22], [25])](image)

A common consensus among behavioral researchers is that the integrating conflict management style leads to cooperation and collaboration between all parties and win-win solutions, which makes it the most appropriate style for managing conflict [2], [6], [15]. However, they also agree that “one style may be more appropriate than another depending upon the situation” [6, p. 425].

The relationship between interpersonal conflict and conflict management styles is a two-way relationship because conflict management styles may be viewed either as antecedents or as consequences of
interpersonal conflict [2]. On the one hand, using different conflict management styles may change the levels of interpersonal conflict; on the other hand, different levels of interpersonal conflict may induce individuals to adopt different conflict management styles [2].

### 3.3. Evolution of software testing

Software “testing is as old as coding” [10], and its evolution is reflected from the changing definition of testing itself. In the early days (1950’s), the activities involved in writing a program were as simple as analysis and coding and testing [28]. Testing, a post-hoc activity that was usually carried out by the same person who did the coding, was to ensure the program could run and also meet expectations.

In his 1979 book [17], *The Art of Software Testing*, Myers defines testing as “the process of executing a program or system with the intent of finding errors.” The focus of software testing was on “finding errors” by using some destruction-oriented means.

Hetzel [11] defines testing as “any activity aimed at evaluating an attribute of a program or system,” and as “the measurement of software quality.” Here the focus of software testing had shifted to product evaluation and quality assurance. However, to better evaluate and assure software quality, we have to carefully choose tools and techniques for measurement and evaluation. These tools and techniques are usually tied to characteristics and attributes of the undertaken program or system.

Caig and Jaskiel [8] define testing as “the concurrent lifecycle process of engineering, using, and maintaining testware in order to measure and improve the quality of software being tested.” This definition of software testing has three major emphases: 1) testing should be concurrent with the development process, not a post-hoc activity after coding, 2) maintaining traceability in testing is critical, and 3) testing is about taking preemptive actions to improve software quality, and quality evaluation alone is not enough.

Today software testing is defined as “a process of verifying and validating that a software application or program 1) meets the business and technical requirements that guide its design and development, and 2) works as expected” [5, p. 2]. According to Bentley, software testing has three main purposes: verification, validation, and defect finding. Verification is to confirm the software meets its technical specifications; validation is to confirm that the software meets the business requirements; and defect finding is to find and fix “important defects” because a tradeoff has to be made between software quality and cost and time to market [5].

Using a four-phase approach, i.e., modeling the software’s environment, selecting test scenarios, running and evaluating test scenarios, and measuring testing progress, Whittaker [34] shows us why eliminating bugs is tricky and why testing is a constant trade-off. He suggests that organizations should 1) “recognize the complex nature of testing and take it seriously,” 2) “hire the smartest people,” and provide “tools and training they need to learn their craft,” and 3) “listen to them when they talk about the software quality.”

### 4. Research model and hypotheses

This section provides a research model (see Figure 2 below) and a list of hypotheses that has been derived from the previous literature review and theoretical basis. Specially, the focus of this study is to examine: 1) the sources of interpersonal conflict in developer and tester relations, 2) the nature and form of such conflict, 3) the impact of such conflict on software development outcomes, and 4) the guidelines and methods for managing such conflict. Software development outcomes include process satisfaction, system quality, system attitude, adherence to budget, adherence to schedule, adherence to specifications, and overall success [2]. Hypotheses related to conflict management styles are advanced for integrating, dominating, and avoiding, however, “no a priori hypotheses are made for accommodating and compromising since findings of past research concerning these two styles are less clear” [2, p. 203].

![Figure 2: Research model](image-url)
Interpersonal conflict on software development outcomes:
H1: Interpersonal conflict will have a negative effect on software development outcomes.
H2: Task conflict will have a positive effect on software development outcomes.
H3: Relationship conflict will have a negative effect on software development outcomes.

Interpersonal conflict and conflict management styles:
H4: Interpersonal conflict will have a negative correlation with integrating and positive correlations with dominating and avoiding.

Conflict management styles on software development outcomes:
H5: Integrating will have a positive effect, while dominating and avoiding will have a negative effect on software development outcomes.

5. Research methodology

This research will be carried out in two phases. In phase one, it will use a qualitative research approach to try to identify the sources of interpersonal conflict between developers and testers. We will interview two key stakeholders in software development, i.e., developers and testers, using open-ended questions. We will ask them to describe their personal conflict experience with other testers or developers. Questions will include: 1) what was the issue? 2) what were the reactions from both conflicting parties? and 3) how did they get the conflict resolved? After finishing the interviews, we will analyze the transcripts of the interviews, and try to identify and organize the sources of conflict between developers and testers in a structured, easy-to-understand, and easy-to-communicate way.

We will then use a quantitative research method for phase two. A survey instrument will be administered to developers and testers in several large software companies. The questionnaire will measure three constructs: interpersonal conflict, conflict management styles, and software development outcomes. The scales to measure these three constructs will be adopted from well-established instruments [2], [14], [23]. We plan to use factor analysis and structural equation modeling to analyze our constructs and to test our hypotheses.

6. Conclusion

This paper has proposed several research questions related to interpersonal conflict between developers and testers in software development. It also reviewed the related concepts, including interpersonal conflict, conflict management styles, and practices of software testing. It advanced a research model and a set of hypotheses, which are to be tested in future empirical studies.

7. References
