January 2014

The Impact of Software Testing Governance Choices

Xihui Zhang
University of North Alabama

Colin Onita
University of Akron

Jasbir Dhaliwal
University of Memphis, jdhaliwl@memphis.edu

Recommended Citation

Xihui Zhang, Colin Onita, and Jasbir Dhaliwal. "The Impact of Software Testing Governance Choices" *Journal of Organizational and End User Computing* 26.1 (2014): 66-85. https://doi.org/10.4018/joeuc.2014010104

development in the form of software development lifecycles which recognized testing as a distinct sequential stage after coding. This led to the growth of software testing as a distinct profession and science – and the emergence of *software development and testing integration* as a crucial organizational IT governance challenge (Zhang et al., 2010). Recent advances in agile methods for both software development and testing (Crispin & Gregory, 2009; Highsmith & Cookburn, 2001; Lee, 2008) have added increased impetus to the need for resolving this challenge. The fact that the proportion of total IT acquisition expenditures that are spent on software testing is going up, because of the increased complexity, application interconnectivity, global-scale, and real-time nature of modern business systems, also calls for an increased focus on this issue as a managerial and theoretical phenomenon in software engineering.

Given the dearth of empirical studies that have explored this phenomenon to provide guidance for industrial practices, software organizations are using a wide diversity of approaches (which are often contradictory) to cope while continuing to make the case that it is a critical area of concern. Consider the following two examples:

1. **Software Testers at Microsoft Corporation:** (a) are not part of a distinct organizational unit for testing, (b) report to the same executives as developers, and (c) are matched to particular developers in agile development teams (Page et al., 2008).
2. **Software Testers at FedEx IT Services:** (a) are part of a distinct organizational unit for testing, (b) report to a different executive than developers, and (c) are not matched to particular developers (Miller, 2009).

While both organizations are known for their innovativeness in the software engineering of business systems, they obviously are using completely contradictory IT governance methods for integrating development and testing. This paper investigates the underlying effectiveness of such IT governance practices for software testing by empirically exploring the organizational, group, and individual impacts of strategic, tactical, and operational software testing governance mechanisms.

The two specific research questions driving this research are:

1. What are the key components of a framework that can guide IT governance decisions pertaining to the integration of software development with testing?
2. What are the empirical impacts of various IT governance mechanisms on organizational, group, and individual level variables pertaining to the integration of software development with testing?

The paper proceeds as follows. The next section tackles the first research question and develops a framework that captures the key dimensions of software testing governance by drawing on the prior literature on both software engineering and IT management. The section after then describes an empirical study that was undertaken to investigate aspects of the framework. This is followed by a section that details our research findings. The section after that not only explores the implications of our findings in relation to both industry best practices and theory development, but also recognizes the limitations of our approach while providing pointers for future research. Finally, the last section provides an overall conclusion.

THEORETICAL DEVELOPMENT

The objective of this study is to examine the impact of the governance of software testing on a set of dependent variables. Specifically, this study explores the impact of three governance mechanisms: the existence of a distinct corporate testing unit, developers and testers reporting to different executives, and one-to-one matching between developers and testers, which are governance mechanisms identified at strategic, tactical, and operational levels, respectively.

The dependent variables too can be classified into three major categories: organizational, group, and individual impacts. Organizational impacts are represented by software quality, value of testing, and development/testing alignment. Group impacts are represented by strategy alignment, capability alignment, and social systems of knowing. Individual impacts are represented by trust between developer and tester, partnership between developer and tester, and job satisfaction. These dependent variables are chosen because they are closely related to the context of software development and testing.

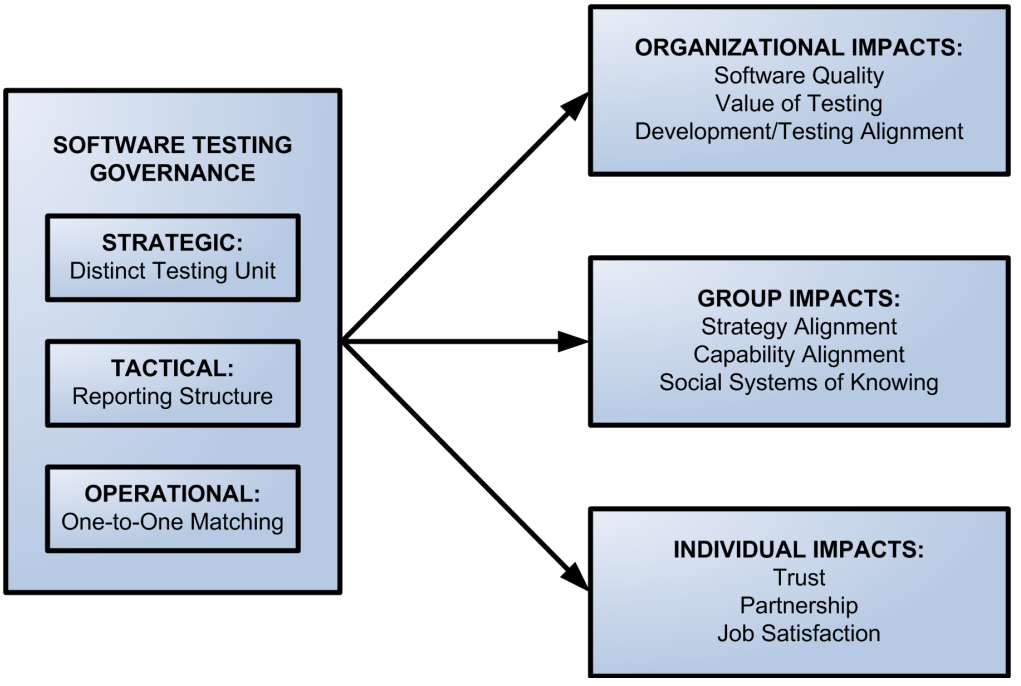
Guided by theory and past research, a framework is proposed which asserts that the existence of a distinct corporate testing unit, developers and testers reporting to different executives, and one-to-one matching between developers and testers will have significant impact on a set of dependent variables (see Figure 1). To simplify the data analysis process, we tested three separate models, each with only one independent variable and the same set of the dependent variables. We provide theoretical support for the hypothesized relationships in the following sections.

Relating IT Governance to Software Testing Governance

Weill and Ross (2004) have demonstrated the criticality of IT governance by showing that firms with better than average governance earn at least 20 percent higher return on assets than organizations with weaker governance. This suggests that it may be worthwhile for software engineering executives to carefully consider a governance perspective to integrating the software development and testing functions.

The literature on IT governance yields several nuanced and related definitions that can be applied to the case of the role of development and testing in software development. Generally, IT governance comprises the leadership, organizational structures and processes that ensure that the organization's IT sustains and

Figure 1. A framework for software testing governance



extends the organization's strategy and objectives (ITGI, 2003; Van Grembergen, 2002). Applying this definition to the case of the integration of development and testing functions in software development, we can define software testing governance as involving the leadership, organizational and integrative processes that ensure the successful implementation of software development strategy. Software testing governance needs to be differentiated from day-to-day software testing management that focuses on *what* specific software testing decisions are being made. Rather, software testing governance is the set of decisions about *who* makes software testing decisions and *how* these decisions are made (Weill, 2004). In other words, it prescribes the structures and processes through which the organization's testing objectives are set, and defines the means for attaining those objectives and monitoring performance.

The IT governance literature emphasizes the importance of the relationship/overlap between corporate/enterprise governance and IT governance and builds upon the former (Luftman & Brier, 1999; Sambamurthy & Zmud, 1999; Weill, 2004). Similarly, our approach involves defining and thinking about software testing governance using the IT governance literature as the base for theory development. As such, software testing governance represents the enterprise's software engineering management system through which its portfolio of software development and testing efforts are directed and controlled. In essence, software testing governance can therefore be viewed as the distribution of software testing decision-making rights and responsibilities among software engineering stakeholders, and the procedures and mechanisms for making and monitoring strategic decisions regarding software testing.

Given that the key issue in software testing governance pertains to its integration with software development, it is also important to consider the relationship between governance and strategic alignment. Webb et al. (2006) have taken such an approach to try amalgamating the range of nuanced definitions for IT governance by proposing the following definition: "IT governance is the strategic alignment of IT with the business such that maximum business value is achieved through the development and maintenance of effective IT control and accountability, performance management, and risk management" (p. 7). Using this approach, software testing governance can be viewed as the strategic integration of testing with development to ensure that the value (quality) in software development can be maximized through the implementation and maintenance of effective control and accountability, performance management, and risk management.

Borrowing from prior IT governance studies by Peterson (2003), Peterson et al. (2002), Weill and Ross (2004), and Van Grembergen et al. (2003), software testing governance can be deployed using a mixture of various *structures, processes, and relational mechanisms*. Petersen (2004) relates these to capabilities in governance and provides examples of structural capabilities, process capabilities, and relational capabilities. De Haes and Van Grembergen (2008, 2009) also utilize this categorization comprising structures, processes, and relational mechanisms for governance. In our view, this categorization can be transposed on the three levels of the organizational management: *strategic, tactical, and operational*. In our model for software testing governance (see Figure 1), structural mechanisms are represented at the strategic level, process mechanisms are represented at the tactical level, and relational mechanisms are represented at the operational level. Thus, strategic structures in software testing governance pertain to institutional issues relating to organizational design that specify the precise formal organizational role of the testing group. Similarly, tactical processes in software testing governance specify controlling, coordinating, and reporting guidelines between testing and development groups. Operational relational mechanisms in software testing governance, however, clarify the participative and collaborative relationships between developers and testers as they work together in software engineering. Relational mechanisms are vital in this software testing governance framework

as they dictate the informal day-to-day working interactions between developers and testers, even when the appropriate formal strategic and tactical structures and processes are in place (Callahan & Keyes, 2003; Keill et al., 2002; Weill & Broadbent, 1998).

Three Levels of Software Testing Governance

Strategic Structures for Software Testing Governance

Prior literature on formal structures for governing software testing is largely non-existent. However, various aspects can be culled from the IT governance literature as being pertinent to the integration of software testing and development. These include: the existence of a distinct organizational unit for software testing, its placing in the organizational hierarchy, formalized strategic steering committees for software engineering management, formal structures for measuring and managing strategic alignment between distinct but related organizational units, and formalized high-level participation on executive committees (De Haes & Van Grembergen, 2008, 2009). Amongst these considerations, the most significant pertains to the existence of a distinct organizational unit for software testing (Miller, 2009). It can be argued that the institutionalization of such a distinct testing unit facilitates formal planning and control governance of software testing. It also promotes the growth of professionalism and identity for the testing group and clarifies the specific focal points for strategic decision making pertaining to budgets, resources, methodologies, and strategic scope of testing. The existence of a distinct organizational unit also facilitates the measurement of return-on-investment and value metrics pertaining to the contribution of the unit at a strategic level. It also provides software testers the opportunity to provide input into strategic organizational deliberations that have the potential of impacting them. The existence of a distinct testing unit also provides the basis for strategic considerations pertaining to centralized, decentralized, and federated governance mechanisms (Sambamurthy & Zmud, 1999) as part of strategic analysis.

Tactical Processes for Software Testing Governance

There is also a dearth of studies that have focused on software testing governance at this level. The general IT governance literature identifies reporting structures, service level agreements, the use of methodologies such as balanced scorecards and COBIT (a framework for IT management and IT governance), and charge-back arrangements as being pertinent (De Haes & Van Grembergen, 2008, 2009). Amongst these, relative reporting structures for development and testing, use of agile versus lifecycle software engineering methodologies, and charge-back arrangements for software testing can be identified as being the most relevant to the integration of software development and testing. Developers reporting to a different manager than testers can be expected to create significant integration and alignment issues as compared to the case where they report to the same manager. The use of agile methodologies (Crispin & Gregory, 2009; Highsmith & Cookburn, 2001; Lee, 2008) for software development is generally associated with having developers and testers report to the same executive such as at Microsoft Corporation (Page et al., 2008). This is because agile processes necessitate frequent and intensive collaboration between developers and testers working together in "scrums" whose work is coordinated in prescribed "sprints" (Larman & Vodde, 2008). Organizations subscribing to the use of systems development lifecycle methodologies can generally be expected to opt for reporting processes where developers report to different managers than testers. The structured stages of the lifecycle, whereby a testing phase generally follows a coding/development phase, facilitate this as prescribed by Teo and King's (1999) notion of sequential integration. Given that units of code are passed over to testers by developers as formalized process handoffs, the

two related activities can be managed using separate reporting mechanisms. Given that the role of the testing function is to verify and validate the work of developers by providing feedback about defects and bugs that are found in testing, chargeback processes, whereby testing costs are "charged" back to development groups, also represent a key governance aspect at this level.

Operational Relational Mechanisms for Software Testing Governance

Significant literature exists in relation to the operational governing mechanisms for software testing. Most of this relates to the measurement and management of conflict between developers and testers (Cohen et al., 2004; Pettichord, 2000; Zhang et al., 2008; Zhang et al., 2013). In addition to this, pertinent aspects that can be culled from the IT governance literature (Dhaliwal et al., 2011; De Haes & Van Grembergen, 2008, 2009; Petersen, 2004) and applied to our context include job rotation, co-location, cross training, knowledge management, as well as formal and informal interactions between developers and testers. Given the relative roles that developers and testers play in software engineering, these can be viewed as being sub-aspects of a higher level construct that can be termed: one-to-one matching between particular developers and testers. A specific tester working on a stable basis to provide defect and quality feedback to a particular developer can be expected over time to yield defined impacts.

Three Levels of Impacts of Software Testing Governance

Following the literature, we chose salient dependent variables that are important in the day-to-day as well as the long-term management of software development organizations, and which are influenced by software testing governance choices. Specifically, our study includes constructs such as partnership that were deemed by Preston and Karahanna (2009) and Luftman and Kempaiah (2007) to be important components of a good IT strategy. Partnership measures the rapport between sub-units and their interaction including issues of trust, shared goals, and values. Value of testing (Luftman & Kempaiah, 2007) deals with perceptions of the benefits of interaction as well as the metrics used to quantify the performance output of a sub-unit and its relative contribution to the other sub-unit's output.

Alignment is another important concept that has been studied in IS literature (Henderson & Venkatraman, 1993; Luftman & Kempaiah 2007; Preston & Krahanna, 2009) and is also important in our list of dependent variables. Henderson and Venkatraman (1993) indentify two main components of alignment – strategy alignment and capability alignment, which we measure as alignment between two individual subunits of the IT department (i.e., development and testing subunits). Following Preston and Karahanna (2009), our paper uses social systems of knowing – defined as the informal interaction between individuals or groups of software developers and testers – as a salient variable that can be influenced by governance choices.

Clearly, the choices for software testing governance can have impacts on an organization as a whole (e.g., software development organizations), groups (e.g., development groups and testing groups), and individuals (e.g., developers and testers). As such, we categorize the dependent variables representing impacts of the choice of software testing governance into three levels, including organizational, group, and individual (see Figure 1). The dependent variables that represent organizational impacts include: software quality, value of testing, and development/testing alignment. The dependent variables that represent group impacts include: strategy alignment, capability alignment, and social systems of knowing. The dependent variables that represent individual impacts include: trust between developer and tester, partnership between developer and tester, and job satisfaction.

Organizational Impacts

For the organizational impacts, we investigate three salient outcomes of governance choices: software quality, value of testing, and development/testing alignment. First we look at the overall software quality as an important organizational outcome. The quality of software developed has important implications to the success of a software development organization. We posit that all three independent variables (i.e., the existence of a distinct corporate testing unit, developers and testers reporting to different executives, and one-to-one matching between developers and testers) will positively influence the quality of the software developed. Having a distinct corporate testing unit allows the testing unit to provide a more cogent and efficient testing strategy and implementation than when testing is only a small part of the development (Miller, 2009). Zhang et al. (2010) pointed out three important advantages with the existence of a distinct corporate testing unit: (1) testers will focus on testing; (2) testers will feel less pressure to ship; and (3) testers will provide "an objective look at the software being tested" (Craig & Jaskiel, p. 297). Myers (2004) argues that "a programming organization should not test its own programs" (p. 16) because development unit and testing unit have distinct objectives. Similarly, developers and testers reporting to different executives creates a stronger testing unit that is better able to both act as a validation entity as well as an improvement entity for the software developed. Finally, having one-to-one matching between developers and testers "facilitates good communication and free flow of information" (Zhang et al., 2010, p. 4); it also has been shown to improve the quality of software by providing immediate and personalized feedback about a piece of software (Page et al., 2008).

H1a: The existence of a distinct corporate unit for software testing will positively influence the quality of software developed.

H2a: Developers and testers reporting to different executives will positively influence the quality of software developed.

H3a: One-to-one matching between developers and testers will positively influence the quality of software developed.

The second organizational impact investigated is the perceived value of testing in the organization (Luftman & Kempaiah, 2007). It is easy to see how both of a distinct corporate testing unit and developers and testers reporting to different executives would improve the perceived value of testing to the organization. Having a clear delineation of departments and responsibility allows the organization to both clearly perceive and quantify the outputs and benefits of testing to the organization. It also allows testing to have a more coherent view of itself and to be more in control of its strategies and capabilities. One-to-one matching pairs up individual developers with individual testers and thus leads to the creation of personal rapport and relationships between testers and developers that lead to a better perception of testers in software development and in the overall organization.

H1b: The existence of a distinct corporate unit for software testing will positively influence organizational understanding of the value provided by testing.

H2b: Developers and testers reporting to different executives will positively influence organizational understanding of the value provided by testing.

H3b: One-to-one matching between developers and testers will positively influence organizational understanding of the value provided by testing.

The third organizational impact investigated pertains to the overall alignment (Henderson & Venkatraman, 1993; Luftman & Kempaiah, 2007; Preston & Karahanna, 2009) between the development and testing units. All three independent variables positively impact the alignment between the software development and testing subunits. The first two variables impact alignment by providing an independent scaffold on which both testing and

development can build their strategies. Since both units are independent of each other, they can build internally the specific capabilities that are required to enact their stated strategies. Also, since both units are sovereign, independent units, they can interact on similar terms and reach a common understanding of software creation goals and strategies. One-to-one matching also leads to better alignment between development and testing due to the increased communication and rapport between individual developers and testers. Since they are in frequent communication and interaction, individual developers and testers are more likely to create a common language and understanding of their jobs (Preston & Karahanna, 2009) as well as be able to know the needs of the other party better.

H1c: The existence of a distinct corporate unit for software testing will positively influence development/testing alignment.
H2c: Developers and testers reporting to different executives will positively influence development/testing alignment.
H3c: One-to-one matching between developers and testers will positively influence development/testing alignment.

Group Impacts

For the group impacts, we investigate three salient outcomes of governance choices: strategy alignment, capability alignment, and social systems of knowing. As part of alignment between developers and testers, we investigate both strategy and capability alignment (Henderson & Venkatraman 1993) of the development and testing groups. These two types of alignment pertain to the strategies and capabilities of each individual group (development and testing) and how the group strategy and capabilities is in alignment or harmony with its counterpart group's strategy and capabilities. Having a distinct testing group and a distinct reporting structure will enable both testing and development to create their own individual strategies as well as internally coherent ways of implementing said strategies by building internal capabilities, deploying the right tools, and employing the correct processes. All these, however, will make it more difficult to achieve either strategy or capability alignment between development and testing groups. One-to-one matching, on the other hand, will improve both the strategy and capability alignment between development and testing groups by creating a common language, rapport, and understanding between individual testers and developers (Preston & Karahanna, 2009).

H1d: The existence of a distinct corporate unit for software testing will negatively influence strategy alignment between developers and testers.
H2d: Developers and testers reporting to different executives will negatively influence strategy alignment between developers and testers.
H3d: One-to-one matching between developers and testers will positively influence strategy alignment between developers and testers.
H1e: The existence of a distinct corporate unit for software testing will negatively influence capability alignment between developers and testers.
H2e: Developers and testers reporting to different executives will negatively influence capability alignment between developers and testers.
H3e: One-to-one matching between developers and testers will positively influence capability alignment between developers and testers.

Social systems of knowing (Preston & Karahanna, 2009) are also influenced by software testing governance choices. Having a distinct testing unit and reporting to different executives will decrease the level of informal interaction between developers and testers, negatively influencing the levels of social systems of knowing. On the other hand, one-to-one matching will positively influence the level of social systems of knowing between developers

and testers since they have a closer and more personal interaction.

H1f: The existence of a distinct corporate unit for software testing will negatively influence social systems of knowing between developers and testers.

H2f: Developers and testers reporting to different executives will negatively influence social systems of knowing between developers and testers.

H3f: One-to-one matching between developers and testers will positively influence social systems of knowing between developers and testers.

Individual Impacts

For the individual impacts, we investigate three salient outcomes of governance choices: trust between developer and tester, partnership between developer and tester, and job satisfaction. Having a distinct corporate testing unit and having developers and testers reporting to different executives would lower the interaction frequency and intensity between developers and testers. As developers and testers are working in different units and reporting to different executives, individual communication and interaction has a more formal structure and will have to navigate through multiple levels of the two departments. This would lower the level of trust and partnership between developer and tester. However, this would increase the level of job satisfaction for both developers and testers because both would feel their importance to the organization as they have distinct units and report to different executives.

One-to-one matching between developer and tester, on the other hand, would increase the interaction frequency and intensity between developers and testers and would be beneficial to both the level of trust and partnership between developer and tester. Since individual developers and individual testers are paired up for the duration of a project, they become closer to each other. This increases the inherent partnership in the pair and would also increase their level of job satisfaction.

H1g: The existence of a distinct corporate unit for software testing will negatively influence level of trust between developers and testers.

H2g: Developers and testers reporting to different executives will negatively influence level of trust between developers and testers.

H3g: One-to-one matching between developers and testers will positively influence level of trust between developers and testers.

H1h: The existence of a distinct corporate unit for software testing will negatively influence partnership between developers and testers.

H2h: Developers and testers reporting to different executives will negatively influence partnership between developers and testers.

H3h: One-to-one matching between developers and testers will positively influence partnership between developers and testers.

H1i: The existence of a distinct corporate unit for software testing will negatively influence job satisfaction.

H2i: Developers and testers reporting to different executives will negatively influence job satisfaction.

H3i: One-to-one matching between developers and testers will positively influence job satisfaction.

RESEARCH METHODOLOGY

Measurement Items

All measurement items for both independent variables and dependent variables were either adapted from existing scales or derived from prior literature. The preliminary instrument was pilot tested for appropriateness and clarity, following Churchill (1979). Specifically, the existence of a distinct corporate testing unit was measured by one item: *Software testing represented an identifiable and distinct organizational unit*. Developers and testers reporting to different executives was measured by one item: *Developers reported to a different executive than testers*. The existence of one-to-one matching

between developers and testers was measured by one item: *Testers were largely assigned to support particular developers*. Respondents were asked to score these three measurement items on 7-point Likert-type scales anchored at (1) = strongly agree and (7) strongly disagree.

Software quality was measured using a six-item scale (see Table 1 for the exact measurement items, same below) adapted from scales developed and validated by Barki and Hartwick (2001), measuring six dimensions of the construct: functionality, reliability, usability, efficiency, maintainability, and portability. This adapted six-item scale is in accordance with software quality measurement scales recommended by Issac et al. (2003) and Ortega et al. (2003). Respondents were asked to score the measurement items on 7-point Likert-type scales anchored at (1) = not at all and (7) = definitely. A seven-item scale was created for the measurement of the construct of value of testing, based on Luftman and Kepaiah's (2007) framework.

The constructs of development/testing alignment (3 items), strategy alignment (3 items), and capability alignment (3 items) were adapted and expanded from Preston and Karahanna (2009) and Henderson and Venkatraman (1993). Items for the measurement of social systems of knowing (3 items) were adapted from Preston and Karahanna (2009). Respondents were asked to score the measurement items on 7-point Likert-type scales anchored at (1) = strongly agree and (7) = strongly disagree.

Trust between developer and tester was measured using a four-item scale adapted from scales developed and validated by Simon and Peterson (2000) and Peterson and Behfar (2003), measuring four aspects of the construct: expectations of truthfulness, certainty of trust, integrity, and living up to one's word. Respondents were asked to score the measurement items on 7-point Likert-type scales anchored at (1) = always and (7) = never. Based on Luftman and Kepaiah's (2007) framework, a four-item scale was created for the measurement of the construct of partnership between developer and tester. Job satisfaction was measured using a five-item scale adapted from scales developed and validated by Wright and Cropanzano (1998), measuring five dimensions of the construct: degree of satisfaction with the work itself, degree of satisfaction with co-workers, degree of satisfaction with the way being supervised, degree of satisfaction with opportunities for promotion, and degree of satisfaction with pay and benefits. Respondents were asked to score the measurement items on 7-point Likert-type scales anchored at (1) = strongly agree and (7) = strongly disagree.

Data Collection

An online survey instrument was then developed, and the survey link was distributed by individual emails to software development professionals. We used "Request for Research Assistance" as the subject for the soliciting emails. In the body of the email, we provided information about the purpose of our study. We also assured recipients that their responses would be kept completely confidential and that there would not be a way for us to link their responses back to them or to their organizations. The respondents were offered as an incentive a summary report of the survey results if requested. A second email, serving as a reminder, was sent three weeks after the first one.

We obtained a total of 1836 unique names and their corresponding emails from three major sources: a database provided by a software testing research center, an online directory of software testers and consultants, and the SourceForge.net portal. All in all, 196 people (10.68%) responded to the online survey. The majority of the respondents were employees of US based organizations with a significant software development and testing employee base (over 140 of the obtained responses). The remainder of the respondents came from the open source portal – SourceForge.net. There is no significant difference between the responses of respondents from SourceForge and the responses of respondents from regular US based organizations. Among them, 46.4% identified themselves as developers, another

Table 1. Constructs and measurement items

CONSTRUCT	MEASUREMENT ITEMS
Distinct Testing Unit	Software testing represented an identifiable and distinct organizational unit
Reporting Structure	Developers reported to a different executive than testers
One-to-One Matching	Testers were largely assigned to support particular developers
Software Quality	• The software developed is reliable (it is always up and running, runs without errors, and does what it is supposed to do). • It is easy to tell whether the software is functioning correctly. • The software can easily be modified to meet changing user requirements. • The software is easy to maintain. • The software is easy to use. • The software performs its functions quickly.
Value of Testing	• There are established testing metrics for demonstrating the value of testing to the organization. • There are established development metrics to demonstrate the value of development to the organization. • The organization uses balanced measurements that are understood and accepted by both development and testing, to measure their relative contributions. • There are explicit service level agreements in place for assessing the contribution of testing to software development. • There are explicit benchmarking standards available for assessing the contribution of the testing group. • There are formal assessments and reviews conducted for evaluating the success of testing efforts. • Continuous improvement processes exist for advancing testing efforts.
Development/Testing Alignment	• The software testing strategy is congruent with the software development strategy in your organization. • Decisions in test planning are tightly linked to decisions in development planning. • Our testing and development strategy are closely aligned.
Strategy Alignment	• The scope of the development group is tightly linked with that of the testing group. • The governance of the development group is in harmony with that of the testing group. • The resources of the development group are aligned with those of the testing group.
Capability Alignment	• The software testing processes is congruent with the software development strategy in your organization. • Our testing infrastructure is tightly integrated with development infrastructure. • Our testing and development capabilities are closely aligned.
Social Systems of Knowing	• Developers have regular informal contact with testers. • Developers regularly socialize with testers outside of the work setting (social gatherings, golf, tennis, etc.). • Developers have regular informal exchanges with testers.
Trust between Developer and Tester	• To what extent were developers and testers truthful to each other? • To what extent could developers and testers trust each other? • To what extent did developers and testers show integrity in their interactions? • To what extent could developers and testers count on the other to live up to their word?
Partnership between Developer and Tester	• The testing leadership plays a direct role in IS development planning. • Testing and development rewards/penalties are based on shared goals and risk factors. • There is a high level of trust between testing and development. • Development and testing commonly partner to sponsor and champion IS initiatives.
Job Satisfaction	• I am satisfied with the work that I do in my job. • I am satisfied with my coworkers. • I am satisfied with the way I am supervised. • I am satisfied with opportunities for promotion in my job. • I am satisfied with my pay and benefits.

42.3% identified themselves as testers, and the remaining 11.2% identified themselves as other software development professionals. Responses were removed from the final data set if (1) they were not from developers or testers, or (2) they contained over 60% of missing values. As a result, a total of 159 responses were included in our data analysis: 80 were from developers, and 79 were from testers.

Demographics of the Respondents

Demographics of the respondents assessed include: gender, education, years of job related work experience, years with current software development and testing organization, and gross annual income (see Table 2). The ratio of male respondents (67.92%) and female respondents (32.08%) was roughly 2:1. More than 80% of the respondents had a bachelor's degree (43.40%) or a master's degree (38.99%) as their highest degree. More than half of them (54.09%) had over 10 years work experience related to their current job, and more than half of them (51.57%) had spent 1 to 5 years with their current software development and testing organization. About 79.25% of the respondents had a gross annual income in the range of $50,000 - $100,000.

DATA ANALYSIS AND FINDINGS

Data Transformation

Before analyzing the data, we transformed all the data items to simplify data analysis and results interpretation. Specifically, we created three new data items (DTU, RS, and OM) from the original three data items for the independent variables, including the existence of a distinct corporate testing unit, developers and testers reporting to different executives, and the existence of one-to-one matching between developers and testers, respectively. For each new data item, we assigned 1 to it if the value associated with the original data item is 1, 2, or 3; we assigned 2 to it if the value associated with the original data item is 5, 6, or 7; and we assigned 3 to it if the value associated with the original data item is 4. These formed three groups: group 1, group 2, and group 3. Data items in group 1 and group 2 were used in the data analysis process, and those in group 3 were not used.

For the dependent variables, the transformation was straightforward. New data items were created, and each had a value that was a simple summation of the values of its associated original data items. For instance, the value of JS (job satisfaction) equals to JS1 + JS2 + JS3 + JS4 + JS5.

Data Analysis

Three independent-samples t tests were performed using SPSS v. 17 for Windows. "Exclude cases listwise" was used for missing values. For each t test, the test variables were the dependent variables, and the grouping variables were DTU, RS, and OM, respectively. Grouping variables were considered to have made a significant difference in test variables if the p-value (strength of significance) was .05 or less (a lower p-value indicates a stronger level of difference). The primary goal herein was to determine whether each of the three independent variables, i.e., the existence of a distinct corporate testing unit, developers and testers reporting to different executives, and the existence of one-to-one matching between developers and testers, influences the dependent variables. We used Cronbach's alpha to show good reliability (above .7, Nunnally, 1978) for all the constructs used in this paper (Job Satisfaction .71; Software Quality .89; Trust .92; Developer Tester Alignment .91; Social Systems of Knowing .80; Value of Testing .89; Partnership .70; Strategic Alignment .89; Capability Alignment .86).

Results

The Existence of a Distinct Corporate Testing Unit

The results of the independent samples t test using the existence of a distinct corporate testing unit (DTU) as the grouping variable are presented in Table 3. There are 138 data points: 118 in group 1 and 20 in group 2. The value of "software quality" reported in group 1 is significantly ($p = .034$) higher than that of group 2. The value of "value of testing" reported in group 1 is (almost) significantly ($p = .054$) lower than that of group 2. The value of "development/testing alignment" reported in group 1 is significantly ($p = .002$) lower than that of group 2.

Table 2. Demographics of the respondents (N = 159)

CATEGORY	VALUE	FREQUENCY	PERCENTAGE
Gender	Male	108	67.92%
	Female	51	32.08%
Education	HS diploma	10	6.29%
	Associate's degree	12	7.55%
	Bachelor's degree	69	43.40%
	Master's degree	62	38.99%
	Doctoral degree	6	3.77%
Years of job related work experience	Less than 1 year	2	1.26%
	1 to 3 years	15	9.43%
	3 to 5 years	11	6.92%
	5 to 7 years	18	11.32%
	7 to 10 years	27	16.98%
	Over 10 years	86	54.09%
Years with current software development and testing organization	Less than 1 year	8	5.03%
	1 to 3 years	39	24.53%
	3 to 5 years	43	27.04%
	5 to 7 years	21	13.21%
	7 to 10 years	22	13.84%
	Over 10 years	26	16.35%
Gross annual income	Under $25,000	9	5.66%
	$25,000 to $50,000	10	6.29%
	$50,000 to $75,000	57	35.85%
	$75,000 to $100,000	69	43.40%
	$100,000 to $125,000	11	6.92%
	Over $125,000	3	1.88%

These results suggest that the formalization of a distinct corporate unit for software testing (group 1) improves: (1) the quality of software developed, (2) organizational understanding of value provided by testing, and (3) development/testing alignment.

Developers and Testers Reporting to Different Executives

The results of the independent samples t test using developers and testers reporting to different executives (RS) as the grouping variable are presented in Table 3. There are 144 data points: 104 in group 1 and 40 in group 2. The value of "strategy alignment" perceived in group 1 is significantly ($p = .008$) higher than that of group 2. The value of "capability alignment" perceived in group 1 is significantly ($p = .009$) higher than that of group 2. The value of "social systems of knowing" perceived in group 1 is significantly ($p = .003$) higher than that of group 2. The value of "trust between developer and tester" perceived in group 1 is significantly ($p = .012$) higher than that of group 2.

Table 3. Independent samples tests with DTU, RS, and OM (Note: Numbers in bold represent significant relationships; numbers in italics represent t values; Numbers in parentheses represent p values)

	INDEPENDENT VARIABLE		
DEPENDENT VARIABLE	**Distinct Testing Unit**	**Reporting Structure**	**One to one Matching**
Organizational Impacts			
Software Quality	***2.138* (.034)**	*-.484* (.629)	***2.118* (.036)**
Value of Testing	***-1.942* (.054)**	*-1.171* (.244)	*.359* (.720)
Development/Testing Alignment	***-3.153* (.002)**	*.996* (.321)	*-.684* (.495)
Group Impacts			
Strategy Alignment	*-1.686* (.095)	***2.716* (.008)**	*-.940* (.350)
Capability Alignment	*-.212* (.833)	***2.643* (.009)**	*.376* (.707)
Social Systems of Knowing	*-1.159* (.249)	***2.993* (.003)**	*-.465* (.643)
Individual Impacts			
Trust between Developer and Tester	*-.484* (.629)	***2.554* (.012)**	*1.036* (.302)
Partnership between Developer and Tester	*-.762* (.447)	*1.110* (.269)	*-1.797* (.075)
Job Satisfaction	*-.202* (.840)	*-.259* (.796)	***-2.626* (.010)**

The above results suggest that developers and testers reporting to different executives (group 1) leads to: (1) reduced strategy alignment, (2) reduced capability alignment, (3) reduced social systems of knowing, and (4) reduced trust between developers and testers.

The Existence of One-to-One Matching Between Developers and Testers

The results of the independent samples t test using the existence of one-to-one matching between developers and testers (OM) as the grouping variable are also presented in Table 3. There are 130 data points: 40 in group 1 and 90 in group 2. The value of "software quality" reported in group 1 is significantly ($p = .036$) higher than that of group 2. The value of "job satisfaction" reported in group 1 is significantly ($p = .010$) lower than that of group 2.

The above results suggest that one-to-one matching between developers and testers (group 1) improves: (1) the quality of software developed, and (2) job satisfaction.

Hypothesis Tests

The hypotheses were assessed by examining t-values and p-values generated from the three independent-samples t tests. The hypothesis test results are summarized in Table 4. Hypotheses H1a, H1b, H1c, H2d, H2e, H2f, H2g, H3a, and H3i are supported, and the remaining hypotheses are not supported. Of all the 27 hypotheses, 9 hypotheses are supported, and 18 hypotheses are not supported. We think the fact that the majority of our hypotheses are not supported is because the scope of the impact of the three independent variables (i.e., the existence of a distinct corporate testing unit, developers and testers reporting to different executives, and one-to-one matching between developers and testers) on the three level of dependent variables (i.e., organizational level, group level, and individual level). If our model and data were perfect, we could argue for the following: Whether or not to have a distinct corporate testing unit is a strategic decision, and this decision will have significant impact on the dependent

variables at the organizational level but not on those at the group level or the individual level. Similarly, whether or not to require developers and testers to report to different executives is a tactical decision, and this decision will have significant impact on the dependent variables at the group level, but not on those at the organizational level or the individual level. And finally, whether or not to match developers and testers one-to-one is an operational decision, and this decision will have significant impact on the dependent variables at the individual level, but not on those at the organizational level or the group level.

DISCUSSION

Implications of Findings

Our research complements current governance research by focusing attention on a somewhat overlooked aspect of the implications of high level governance choices regarding software development and testing departments on the result of the software development and testing process, as well as on the internal interactions between software developers and testers. This paper offers insights on how governance choices regarding the departmental make-up, reporting structure, and the internal software development and testing team composition impact the quality of the software produced, job satisfaction, alignment between software developers and testers as well as the personal relationship between developers and testers. These components have been looked at before, but not in the context of the software development and testing departments. Some of the variables investigated by this research are peculiar to software development and testing departments (one to one testing, independent testing unit, software quality, etc.); and this paper breaks ground in showing that governance choices can have significant impacts on the quality of the output and relationships in software development and testing departments.

CIOs and senior IT executives who manage software development have to make three important governance choices. At the strategic level, a decision has to be made about whether to create a distinct organizational unit for testing. At the tactical level, a decision has to be made about whether to let testers report to the same executive as developers. At the operational level, a decision has to be made about whether to closely match individual testers to designated developers.

Among these three governance choices, the first is of the most significance and should be the focus of governance deliberations. This is because this governance choice has significant organizational impacts such as increased software quality, increased organizational understanding of value provided by testing, and increased alignment between development and testing. Putting testers in a distinct organizational unit provides a host of positive impacts and has no significant detrimental effects on our investigated outcome variables.

Similarly, IT leaders may want to prioritize one-to-one matching between developers and testers in their governance choices because it also yields a variety of positive impacts, at both organizational and individual levels. These benefits include increased software quality and increased job satisfaction.

IT leaders who are structuring reporting mechanisms for testers and developers need to understand that having them report to different executives has multiple negative consequences such as decreased strategy alignment, decreased capability alignment, decreased social systems of knowing, and decreased trust between developers and testers.

A key theoretical implication of our result is that the goals of the organization should drive governance design choices. For instance, if the particular goal is to improve software quality, then the focus should be on setting up distinct testing units and one-to-one matching mechanisms as against decisions pertaining to the reporting structure.

Table 4. Summary of hypothesis tests

HYPOTHESIS	T-VALUE	P-VALUE	SUPPORT FOR HYPOTHESIS
H1a: The existence of a distinct corporate unit for software testing will positively influence the quality of software developed.	2.138	.034	Supported
H1b: The existence of a distinct corporate unit for software testing will positively influence organizational understanding of the value provided by testing.	-1.942	.054	Supported
H1c: The existence of a distinct corporate unit for software testing will positively influence development/ testing alignment.	-3.153	.002	Supported
H1d: The existence of a distinct corporate unit for software testing will negatively influence strategy alignment between developers and testers.	-1.686	.095	Not Supported
H1e: The existence of a distinct corporate unit for software testing will negatively influence capability alignment between developers and testers.	-.212	.833	Not supported
H1f: The existence of a distinct corporate unit for software testing will negatively influence social systems of knowing between developers and testers.	-1.159	.249	Not supported
H1g: The existence of a distinct corporate unit for software testing will negatively influence level of trust between developers and testers.	-.484	.629	Not supported
H1h: The existence of a distinct corporate unit for software testing will negatively influence partnership between developers and testers.	-.762	.447	Not supported
H1i: The existence of a distinct corporate unit for software testing will negatively influence job satisfaction.	-.202	.840	Not supported
H2a: Developers and testers reporting to different executives will positively influence the quality of software developed.	-.484	.629	Not supported
H2b: Developers and testers reporting to different executives will positively influence organizational understanding of the value provided by testing.	-1.171	.244	Not Supported
H2c: Developers and testers reporting to different executives will positively influence development/testing alignment.	.996	.321	Not supported
H2d: Developers and testers reporting to different executives will negatively influence strategy alignment between developers and testers.	2.716	.008	Supported
H2e: Developers and testers reporting to different executives will negatively influence capability alignment between developers and testers.	2.643	.009	Supported
H2f: Developers and testers reporting to different executives will negatively influence social systems of knowing between developers and testers.	2.993	.003	Supported

continued on following page

Table 4. Continued

HYPOTHESIS	T-VALUE	P-VALUE	SUPPORT FOR HYPOTHESIS
H2g: Developers and testers reporting to different executives will negatively influence level of trust between developers and testers.	2.554	.012	Supported
H2h: Developers and testers reporting to different executives will negatively influence partnership between developers and testers.	1.110	.269	Not supported
H2i: Developers and testers reporting to different executives will negatively influence job satisfaction.	-.259	.796	Not supported
H3a: One-to-one matching between developers and testers will positively influence the quality of software developed.	2.118	.036	Supported
H3b: One-to-one matching between developers and testers will positively influence organizational understanding of the value provided by testing.	.359	.720	Not supported
H3c: One-to-one matching between developers and testers will positively influence development/testing alignment.	-.684	.495	Not supported
H3d: One-to-one matching between developers and testers will positively influence strategy alignment between developers and testers.	-.940	.350	Not supported
H3e: One-to-one matching between developers and testers will positively influence capability alignment between developers and testers.	.376	.707	Not supported
H3f: One-to-one matching between developers and testers will positively influence social systems of knowing between developers and testers.	-.465	.643	Not supported
H3g: One-to-one matching between developers and testers will positively influence level of trust between developers and testers.	1.036	.302	Not supported
H3h: One-to-one matching between developers and testers will positively influence partnership between developers and testers.	-1.797	.075	Not Supported
H3i: One-to-one matching between developers and testers will positively influence job satisfaction.	-2.626	.010	Supported

Limitations and Suggestions for Future Research

This study has several limitations. First, each of the three independent variables was measured by a single item. This can be troublesome in survey research such as ours. Future research is thus encouraged to develop and validate multi-item scales for these constructs. For example, further refinement of the one-to-one matching construct between developers and testers is a viable area for new and follow-up studies. Second, software quality was measured by a survey of software developers and testers. It is understandable that there may be a difference between software quality perceived by developers and testers and that perceived by end users. Future research can focus on end users to measure the perceptions of software quality.

Several other directions for future research can also be suggested. First, other researchers may want to explore our constructs in other contexts besides those that drove our study. Second, future studies may want to focus on directly integrating some of the significant relationships identified in this paper into actionable contingency theories. Third, future research can also attempt to triangulate our findings by conducting focused qualitative studies to add another level of validation. Fourth, future work can further refine our framework by including additional theoretically driven antecedents and outcomes. Fifth, future research can investigate whether the organization type and the choices the organization makes regarding their software and testing units moderate any of our proposed relationships.

CONCLUSION

This study investigates the influence of software testing governance choices on a set of dependent variables that represent impacts at the organizational, group, and individual levels. A key conclusion arising from our study is that software testing governance design is a complex task involving the consideration of a broad array of strategic, tactical, and operational choices and a diverse set of organizational, group, and individual impacts. This suggests that context-driven contingency theories may be more appropriate than singular theoretic formulations that focus on narrow imperatives. Our study helps focus managerial attention on the pertinent contextual impacts and relative balance between governance decisional choices. Specifically, our results suggest that, to maximize the benefits resulted from the software testing governance choices, software development organizations should do the following: (1) create a distinct organizational unit for testers, (2) let both development and testing groups report to the same executive, and (3) emphasize one-to-one matching between developers and testers.

REFERENCES

Barki, H., & Hartwick, J. (2001). Interpersonal conflict and its management in information system development. *Management Information Systems Quarterly*, *25*(2), 195–228. doi:10.2307/3250929

Callahan, J., & Keyes, D. (2003). The evolution of IT governance at NB power. In W. Van Grembergen (Ed.), *Strategies for information technology governance*. Hershey, PA: IGI Global. doi:10.4018/978-1-59140-140-7.ch013

Churchill, G. A. (1979). A paradigm for developing better measures of marketing constructs. *JMR, Journal of Marketing Research*, *16*(1), 64–73. doi:10.2307/3150876

Cohen, C. F., Birkin, S. J., Garfield, M. J., & Webb, H. W. (2004). Management conflict in software testing. *Communications of the ACM*, *47*(1), 76–81. doi:10.1145/962081.962083

Craig, R. D., & Jaskiel, S. P. (2002). *Systematic software testing*. Norwood, MA: Artech House Publishers.

Crispin, L., & Gregory, J. (2009). *Agile testing: A practical guide for testers and agile teams*. Boston, MA: Addison-Wesley.

De Haes, S., & Van Grembergen, W. (2008). An exploratory study into the design of an IT governance minimum baseline through Delphi research. *Communications of the Association for Information Systems*, *22*, 443–458.

De Haes, S., & Van Grembergen, W. (2009). An exploratory study into IT governance implementations and its impact on business/IT alignment. *Information Systems Management*, *26*(2), 123–137. doi:10.1080/10580530902794786

Dhaliwal, J., Onita, C., Poston, R., & Zhang, X. (2011). Alignment within the software development unit: Assessing structural and relational dimensions between developers and testers. *The Journal of Strategic Information Systems*, *20*(4), 323–342. doi:10.1016/j.jsis.2011.03.001

Henderson, J. C., & Venkatraman, N. (1993). Strategic alignment: Leveraging information technology for transforming organizations. *IBM Systems Journal*, *32*(1), 4–16. doi:10.1147/sj.382.0472

Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *IEEE Computer*, *34*(9), 120–122. doi:10.1109/2.947100

Issac, G., Rajendran, C., & Anantharaman, R. N. (2003). Determinants of software quality: Customer's perspective. *TQM & Business Excellence*, *14*(9), 1053–1070. doi:10.1080/1478336032000090950

ITGI. (2003). *Board briefing on IT governance* (2nd ed.). Retrieved from http://www.itgi.org

Jehn, K. A., & Mannix, E. A. (2001). The dynamic nature of conflict: A longitudinal study of intragroup conflict and group performance. *Academy of Management Journal*, *44*(2), 238–251. doi:10.2307/3069453

Keill, M., Tiwana, A., & Bush, A. (2002). Reconciling user and project manager perceptions of IT project risk: A Delphi study. *Information Systems Journal*, *12*(2), 103–119. doi:10.1046/j.1365-2575.2002.00121.x

Larman, C., & Vodde, B. (2008). *Scaling lean & agile development: Thinking and organizational tools for large-scale scrum*. Boston, MA: Addison-Wesley.

Lee, E. C. (2008). Forming to performing: Transitioning large-scale project into agile. *Proceedings of AGILE 2008*, Toronto, Canada (pp. 106-111).

Luftman, J., & Brier, T. (1999). Achieving and sustaining business-IT alignment. *California Management Review*, *42*(1), 109–122. doi:10.2307/41166021

Luftman, J., & Kempaiah, R. (2007). An update on business-IT alignment: "A line" has been drawn. *MIS Quarterly Executive*, *6*(3), 165–177.

Miller, D. (2009). Keynote speech: Innovations and best practices in software quality assurance. *The Malaysian Software Testing Board (MSTB) Software Testing Conference (SOFTEC 2009)*, Kuala Lumpur, Malaysia.

Myers, G. J. (2004). *The art of software testing* (T. Badgett, T. M. Thomas, & C. Sandler, Eds.). 2nd ed.). Hoboken, NJ: John Wiley & Sons.

Ortega, M., Pérez, M., & Rojas, T. (2003). Construction of a systemic quality model for evaluating a software product. *Software Quality Journal*, *11*(3), 219–242. doi:10.1023/A:1025166710988

Page, A., Johnston, K., & Rollison, B. J. (2008). *How we test software at Microsoft*. Redmond, WA: Microsoft Press.

Peterson, R. (2003). Information strategies and tactics for information technology governance. In W. Van Grembergen (Ed.), *Strategies for information technology governance*. Hershey, PA: IGI Global. doi:10.4018/978-1-59140-140-7.ch002

Peterson, R. (2004). Crafting information technology governance. *Information Systems Management*, *21*(4), 7–22. doi:10.1201/1078/44705.21.4.20040901/84183.2

Peterson, R., Parker, M. M., & Ribbers, P. (2002, December 15–18). Information technology governance processes under environmental dynamism: Investigating competing theories of decision making and knowledge sharing. In *Proceedings of the 23th International Conference on Information Systems (ICIS)*, Barcelona, Spain.

Peterson, R. S., & Behfar, K. J. (2003). The dynamic relationship between performance feedback, trust, and conflict in groups: A longitudinal study. *Organizational Behavior and Human Decision Processes*, *92*(1), 102–112. doi:10.1016/S0749-5978(03)00090-6

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.

Preston, D. S., & Karahanna, E. (2009). Antecedents of IS strategic alignment: A nomological Network. *Information Systems Research*, *20*(2), 159–179. doi:10.1287/isre.1070.0159

Royce, W. W. (1970). Managing the development of large software systems. In *Proceedings of IEEE WESCON (WESCON 1970)*, Los Angeles, CA (pp. 1-9).

Sambamurthy, V., & Zmud, R. W. (1999). Arrangements for information technology governance: A theory of multiple contingencies. *Management Information Systems Quarterly*, *23*(2), 261–290. doi:10.2307/249754

Simons, T. L., & Peterson, R. S. (2000). Task conflict and relationship conflict in top management teams: The pivotal role of intragroup trust. *The Journal of Applied Psychology*, *85*(1), 102–111. doi:10.1037/0021-9010.85.1.102 PMID:10740960

Teo, T. S. H., & King, W. (1999). An empirical study of the impacts of integrating business planning and information systems planning. *European Journal of Information Systems*, *8*(3), 200–210. doi:10.1057/palgrave.ejis.3000334

Van Grembergen, W. (2002, January 7-10). Introduction to the minitrack: IT governance and its mechanisms. In *Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS)*, Big Island, HI.

Van Grembergen, W., De Haes, S., & Guldentops, E. (2003). Structures, processes and relational mechanisms for IT governance. In Van Grembergen (Ed.), Strategies for information technology governance. Hershey, PA: IGI Global.

Webb, P., Pollard, C., & Ridley, G. (2006, January 4-7). Attempting to define IT governance: Wisdom or folly? In *Proceedings of the 39th Hawaii International Conference on Systems Sciences (HICSS)*, Kauai, HI.

Weill, P., & Broadbent (1998). *Leveraging the new infrastructure: How market leaders capitalize on information technology*. Boston, MA: Harvard Business School Press.

Weill, P. (2004). Don't just lead govern: How top-performing firms govern IT. *MIS Quarterly Executive*, *3*(1), 1–17.

Weill, P., & Ross, J. (2004). *IT governance: How top performers manage IT decision rights for superior results*. Boston, MA: Harvard Business School Press.

Wright, T. A., & Cropanzano, R. (1998). Emotional exhaustion as a predictor of job performance and voluntary turnover. *The Journal of Applied Psychology*, *83*(3), 486–493. doi:10.1037/0021-9010.83.3.486 PMID:9648526

Zhang, X., Dhaliwal, J. S., & Gillenson, M. L. (2010). Organizing software testing for improved quality and satisfaction. *Journal of Information Technology Management*, *21*(4), 1–12.

Zhang, X., Dhaliwal, J. S., Gillenson, M. L., & Moeller, G. (2008). Sources of conflict between developers and testers in software development. In *Proceedings of 14th Americas Conference on Information Systems (AMCIS 2008)*, Toronto, Canada (Paper 313, pp. 1-12).

Zhang, X., Dhaliwal, J. S., Gillenson, M. L., & Stafford, T. F. (2013). The impact of conflict judgments between developers and testers in software development. *Journal of Database Management*, *24*(4), 26-50.

Zhang, X., Hu, T., Dai, H., & Li, X. (2010). Software development methodologies, trends and implications: A testing centric view. *Information Technology Journal*, *9*(8), 1747–1753. doi:10.3923/itj.2010.1747.1753

Xihui Zhang is an Associate Professor of Computer Information Systems in the College of Business of the University of North Alabama. He earned a Ph.D. in Business Administration with a concentration in Management Information Systems from the University of Memphis. His teaching and research interests include the human, social, and organizational aspects of Information Systems. He has published in such leading journals as the Journal of Strategic Information Systems, Information & Management, *and* Journal of Database Management.

Colin G. Onita is an Assistant Professor of Accounting Information Systems in the George Daverio School of Accountancy at the University of Akron. His research deals with IS strategic and tactical alignment, online social networks, IS services evaluation, and medical information source choice and usage issues. He holds a Ph.D. in Business Administration with a concentration in Management Information Systems from the University of Memphis.

Jasbir S. Dhaliwal is Larry W. Papasan Professor of Information Systems at the University of Memphis where he also serves as Associate Dean for Research and Academic Programs. He is the founding Director of the Systems Testing Excellence Program at the FedEx Institute of Technology whose mission is to advance the science of testing through cutting-edge research to provide a stronger theoretical base for industry best-practices. He holds a Ph.D. from the University of British Columbia and has prior academic experience from universities in Singapore, Canada, and Norway.