January 2011

Alignment within the Corporate IT Unit: An Analysis of Software Testing and Development

Colin Onita
University of Memphis

Jasbir Dhaliwal
University of Memphis

Recommended Citation

Colin Onita and Jasbir Dhaliwal. "Alignment within the Corporate IT Unit: An Analysis of Software Testing and Development" *European Journal of Information Systems* 20.1 (2011): 48-66. https://doi.org/10.1057/ejis.2010.52

Alignment within the Corporate IT Unit: An Analysis of Software Testing and Development

Abstract

Strategic alignment between an organization's business strategy/capabilities and those of its information technology (IT) unit is an extensively researched subject that addresses the issue of fit between business and technology strategies. A key gap in the literature is lack of recognition that underlying this macro level of alignment are other more granular levels of alignment involving the interdependent subunits within the corporate IT unit. Given the critical interdependencies between development and testing subunits in software engineering, this paper focuses on an alignment model for ensuring that these two functions work together effectively. A development-testing alignment (DTA) model is described, and a case study investigating its value and application is presented. This development-testing alignment is decomposed into distinct components for the purposes of theoretical clarity and pragmatic application. The case study analysis uses the model to understand and interpret developer-tester alignment in a Fortune 500 company. We found that the development and testing functions were significantly misaligned, and our model identified close to twenty specific aspects that needed to be considered to enhance alignment. These included changes in specificity of scope, governance, resource availability, competencies, and processes. Our analysis shows that the DTA model can be usefully applied for the purpose of understanding tactical alignment between subunits within a corporate IT unit. It also demonstrates that there is value in considering alignment as a dynamic, context-driven, social phenomenon as well as a useful interpretative lens for exploring organizational interactions and interdependencies.

ACM Categories: D.2.5 Testing and Debugging, D.2.9 Management, J.4 Social and Behavioral Sciences, K.6.1 Project and People Management

Keywords: software development, software testing, strategic alignment, case study

Introduction

According to Gartner (2003), on average only 7% of software functionality that was paid for is actually used, with 85% of information systems (IS) projects failing to meet objectives (and 32% being cancelled outright).

As a result of these failures, software testing is becoming an increasingly important activity in software engineering. However, testing is difficult to do and its relationship to development remains tenuous and important for avoiding software failure (Elbert & Dumnke 2007; Page, Johnston & Rollinson, 2009). Development is defined as encompassing all non-testing activities in software engineering, involving business and systems analysts, designers, and coders. Software testing is the process used to identify the correctness, completeness, security, and quality of computer software, including late-stage system testing conducted on a complete, integrated system to evaluate its compliance with specified requirements (Black, 2002). This paper construes testing as all activities and actions (automated or otherwise) taken to ensure that systems are valid in relation to the real world that they model, verified in relation to requirements, and free of all types of errors. Software engineering can fail because of the lack of proper testing and effective collaborative mechanisms between the development and testing functions. Practitioners have also argued that software testing practices remain underdeveloped compared to development practices, and that coordination between testing and development is a key IS management concern (Miller, 2007). These misalignments or gaps often lead to conflict between testers and developers (Pettichord, 2000; Cohen et al., 2004; Ji et al., 2005; Zhang et al., 2008). Most research on software testing deals with technical issues related to code testing (e.g. Vagoun & Hevner, 1997) and other aspects such as software walkthroughs and inspections as specified by IEEE Standard 1028 (1997). A model that focuses on aligning development and testing at all stages of system building and that ties development strategies and capabilities to testing strategies and capabilities within the corporate IT environment would allow managers to better understand and manage their software engineering activities. It would also provide academia with a more structured way to explore and interpret alignment as an organizational issue in software engineering. In this paper we consider such a model and investigate its applicability by way of a case study. In our view, there is a strong case for focusing on the alignment between the development and testing functions in software engineering.

Strategic information technology alignment arises when the business goals and activities of an organization are in harmony with the information systems that support them (McKeen & Smith, 2003). While achieving such alignment has been the focus of research and practice for nearly two decades (Brancheau et al.,

1996; Watson et al., 1997), it still remains a pertinent issue for IT executives, who have consistently ranked strategic alignment as a top priority (Talon et al., 2000; Reich & Nelson, 2003; Luftman & MacLean, 2004; Luftman & Kempiah, 2007; Preston & Karahanna, 2009). Despite the prominence of this issue, alignment within corporate IT units has not been considered in the practitioner and academic literatures. We posit that without proper alignment between the internal elements of a corporate IT unit (such as development and testing), the alignment between business and IT will be tenuous at best.

Researchers have studied alignment models and measurements in other related contexts at the macro business strategy level, investigating alignment issues between business and IT units (Henderson & Venkatraman, 1993; Reich & Benbasat, 2000; Sabherwal et al., 2001). To our knowledge no other paper has attempted to apply alignment concepts at the micro level, within a corporate IT unit and, specifically, to development and testing groups. In our view addressing this gap in the literature may open up beneficial areas of research and practical applications, and it is the *first goal* of this study. Prior research has treated strategic business–IT alignment at a more conceptual level and rationalizes it as a desirable organizational objective. Our study argues that such alignment must provide for more precise lower-level analysis and deconstruction for it to better serve as a basis for managerial learning and action. Therefore, a *second goal* is to use alignment as a theoretical lens for exploring context-driven organic phenomena to generate insight/understanding about discontinuities and unanticipated trajectories in IT management.

Strategic business-IT alignment has been largely studied empirically using cross-sectional data analysis (Preston & Karahanna, 2009). Few interpretative action research or case analyses investigating the practical application of alignment's concepts are available (e.g. Lesca & Caron-Fasan, 2008). Therefore, a *third goal* of our research is to empirically link alignment to the essential management challenges faced by technology organizations and to provide a framework that facilitates managerial understanding and response.

The paper is structured as follows. The next section presents a theoretical model for the alignment of testing and development based on the prior research on strategic alignment. Next, we present methodological issues pertaining to a case study investigation that utilized our model to generate interpretive findings from the research context. We then present and discusses these findings. The last section of the paper presents our conclusions, the limitations of our study, and directions for future research.

Literature Review and Theoretical Development

Both the academic and practitioner literature have indentified developer-tester relations as a key concern in software engineering today (Pettichord, 2000; Cohen et al., 2004; Ji et al., 2005; Elbert & Dumnke 2007, Zhang et al., 2008; Page, Johnston & Rollinson, 2009) . Hayes (2003) finds that more than 50% of surveyed organizations are focusing on developer-tester issues early in the software development lifecycle. Giambastini (2009) argues that for many programs today, more than 50% of the development schedule is devoted to testing and this percentage is increasing. Hutcheson (2003) argues that the growing popularity of rapid application development, agile development methodologies and the push to be first to market with new software has weakened software quality and fractured the relationship between development and testing groups.

The strategic alignment model (SAM) proposed by Henderson and Venkatraman (1993) looks at connections and linkages between business and IT. This model has been heavily modified, most recently by Preston and Karahanna (2009), who bring in ideas about social interactions and shared language to explain how strategic alignment works at multiple levels. Maes et al. (2000) and Avison et al. (2004) also operationalize SAM through an enterprise architecture knowledge management framework to further develop the Henderson and Venkatraman (1993) model. Other research identifies factors enabling and inhibiting strategic alignment (Luftman et al., 1999; Hussin et al., 2002), or tries to measure alignment by looking at internal consistency and external validity (Reich & Benbasat, 1996).

Although a host of conceptual and empirical studies have investigated the strategic alignment concept, little consensus on a definition has emerged. It is our belief that the cause of this definitional chaos is the mistaken assumption that alignment is an objective, measurable phenomenon independent of organizational context. The early 2000s saw a shift in the focus of the alignment literature from “structural/intellectual” constructs to “social/relational” constructs (Reich & Benbasat, 2000; Preston & Karahanna, 2009) that recognizes that IT management is a social activity. However, this “sociality” of alignment has not been studied at the individual organization level where these relational aspects unfold. Our approach is to consider alignment in IT management at a more subtle level, incorporating subjective epistemology and the stance that the reality of organizational alignment is socially constructed. It is therefore important that alignment be considered at a less strategic and deeper analytic level where its sub-constructs are explored in relation to diverse meanings and understandings of complex relationships, interactions, and social construction in the context of an information systems organization.

The definitional subjectivity of strategic alignment as a viable managerial approach can represent a strength if it leads to adaption in its application based on contextually driven understandings of its inherent notions of fit, harmony, congruence, shared meaning, coordination, correspondence, and matching.

Another gap in the prior research on alignment is an exclusive focus on the value of alignment as a strategic construct. The assumption is that unless IT strategy can be aligned to business strategy, the information systems function will not be able to make a useful contribution. In our view, this strategic focus should be expanded to also include tactical elements of alignment such as those presented in this paper. The alignment "lens" can inform IS management at other potentially useful levels of focus and can enhance tactical information systems management in useful ways by facilitating interpretive analysis and situated inquiry of organizational issues, problems, and miscommunications.

The literature can also benefit from more granular comparisons between strategies that are supposed to be aligned for success. Strategic plans that are fraught with internal contradictions and that encompass incoherent sub-aspects can be problematic to measurements of strategic alignment. Focused contextual analysis of the cohesiveness of strategies can help the practical application of the alignment approach. This need for a more granular analysis of cohesiveness also applies to organizational capabilities. Capabilities that are not cohesive can negatively impact strategic alignment.

Current research also ignores alignment issues within corporate IT units and assumes harmonious internal operation among subunits. In many organizations, though, both strategic and functional gaps exist between the development and testing groups, as well as between individual testers and developers (Pettichord, 2000; Cohen et al., 2004; Ji et al., 2005; Zhang et al., 2008). To bridge these gaps, this paper proposes a developer-tester alignment model that draws upon strategic alignment concepts. *Alignment between development and testing strategy/capabilities can be defined as the strategic and operational fit between a firm's development and testing functions.* Since systems development and testing are integral parts of a corporate technology acquisition strategy, they have to be aligned to ensure business success. This DTA focuses on the **fit** between the development and testing subunits and how they operate collaboratively to support each other to achieve the goals of a corporate IT unit.

Conceptualizations of Alignment as Fit

Venkatraman (1989) proposed six conceptualizations of alignment – alignment as moderation, mediation, matching, gestalts, profile deviation, and co-variation. This paper focuses on two of these conceptualizations – matching and co-variation – which have been identified in the strategic IS literature as especially appropriate for IT contexts (Teo & King, 1996; Preston & Karahanna, 2009). They highlight the roles that business and IT play within an organization and clearly depict the relation between business and IT. From a theoretical perspective, these conceptualizations are well suited to our research focus on intra-organizational IT subunits. Furthermore, a third perspective - the coordination perspective – which has not been considered by prior alignment studies, argues that alignment is a function of the interdependencies between two groups. Malone and Crowston (1991) propose a model of coordination based on the characteristics of the interactions between actors. According to their model, collaboration/coordination occurs among actors who share interdependencies and undertake activities to meet goals. The development and testing functions are highly interdependent by their very nature, necessitating significant coordination between them. All three conceptualizations can be adapted to the context of development and testing groups as follows:

- DTA as Matching: This fit is the correspondence or equivalence between development and testing strategies and capabilities. For example, a close mirroring of capabilities, tools, and resources in both the development and testing functions would suggest strong DTA leading to successful implementation of business systems.
- DTA as Co-variation: This fit between development and testing is seen as a co-variation of attributes that characterize each separate function. We can look at the attributes of development and testing separately and investigate how they covary or diverge. For example, if the resource endowment of a development group increases, while that of a testing group decreases, potential misalignment may be indicated.
- DTA as Coordination: This fit focuses on the required coordination between two interdependent functions. Depending on the nature of interdependence, varying coordination mechanisms are appropriate to facilitate collaboration. For example, if testing uses group/team–based

coordination, which is characteristic of a flexible situation, while development uses rule-based coordination, misalignment may exist between the groups

Theoretical Model of Developer-Tester Alignment

Building on the above, this paper proposes an integrated model for applying the salient concepts of alignment to testing and development functions. Figure 1 details the key *structural and flow components* of our DTA model. Structural components are represented by the four quadrants of Figure 1. The left column of Figure 1 represents the strategies and capabilities of a development subunit, and the right column represents the strategies and capabilities of a testing subunit. The DTA model also exhibits four flow components: (1) strategic alignment, (2) capabilities alignment, (3) execution alignment and (4) cohesiveness of each structural component. These flow components bridge the four structural components depicted by the four quadrants of Figure 1.

The first structural component, development strategy (upper left quadrant in Figure 1), encompasses the scope, governance and resource dimensions of the development subunit's strategic planning. The scope of development is defined in terms of specific development goals and allocation of responsibility. Governance involves both the structures and processes used for organizing and controlling the functional unit as well as buy-or-build decisions. Resources are the budgetary and other endowments available to the functional unit. The second structural component, development capabilities (lower left quadrant in Figure 1), deals with the processes, skills, and architecture that impact the success of the development subunit in executing its stated strategies. This structural component includes the applications portfolio being developed, decisions about development models such as the system development life cycle (SDLC), process, skills and architecture. Processes relate to all sequential and parallel activities a functional unit engages in to carry out its mission. Skills pertain to the unit's knowledge and competencies. Architecture comprises the tools, techniques, and methodologies the unit uses as part of its work.

The third structural component, testing strategy (upper right quadrant of Figure 1), includes testing governance, resource allocation strategies, and the scope of testing activities. Decisions about in-house or outsourced testing are considered here, as well as decisions about the structure and governance of a testing unit. The fourth structural component (lower right quadrant of Figure 1) comprises software testing capabilities. The specific processes of testing (traditional, V-model, iterative, agile), as well as specific choices about testing tools,

architecture, communication structure, etc. are considered from an alignment perspective. The individual skills of personnel required for testing are considered.

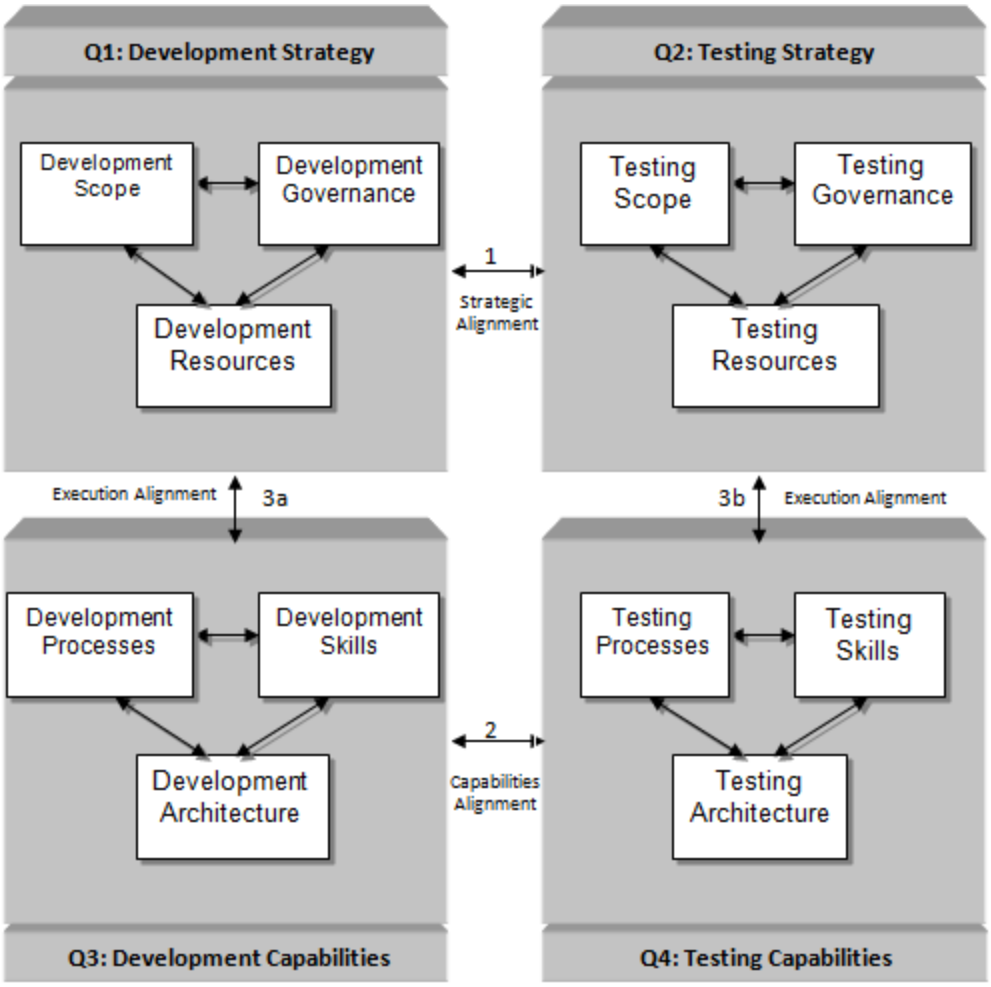

Figure 1: Alignment Model for Development and Testing
(adapted from Henderson and Venkatraman, 1993)

To achieve alignment, all four structural components have to be in harmony. This does not mean that they have to be similarly matched, but that testing complements development and enables development success by providing verification, validation, and bug-finding services. Our model for aligning these subunits within a corporate IT unit differs in an important way from the Henderson and Venkatraman (1993) conceptualization. Our model uses the strategy versus capabilities distinction instead of their internal versus external distinction. This better reflects the resource-based theoretic view of strategic management. We have also changed the focus of

the model from a high-level perspective to a more granular one which is more appropriate to our ***within*** **IT** unit environment.

The four flow components, represented by the arrows in Figure 1 are also critical to understanding and applying the alignment concepts. The three between flow components (as represented by the numbered vertical and horizontal arrows in Figure 1) are strategic alignment, capabilities alignment, and execution alignment. The last internal flow component is cohesiveness (unnumbered arrows within each quadrant) which pertains to the internal consistency of each strategy or capability structural component. These flow components are synthesized from the body of work on strategic alignment (e.g., Henderson & Venkatraman, 1993; Luftman & Kepaiah, 2007; Preston & Karahanna, 2009) that has laid out an understanding of how various organizational, interpersonal, social, and knowledge aspects influence fit between business and the IT service functions. These flow components are examined in the following subsections and are depicted in Table 1.

Developer-Tester Alignment (DTA)	
Cohesiveness	Cohesiveness of Development Strategy (arrows in quadrant 1)
	Cohesiveness of Testing Strategy (arrows in quadrant 2)
	Cohesiveness of Development Capability (arrows in quadrant 3)
	Cohesiveness of Development Capability (arrows in quadrant 4)
Strategy Alignment of Development and Testing (arrow 1)	
Capabilities Alignment of Development and Testing (arrow 2)	
Execution Alignment	Execution Alignment of Development (arrow 3a)
	Execution Alignment of Testing (arrow 3b)

Table 1: Flow Components of Developer-Tester Alignment

Cohesiveness

We considered the cohesiveness of development and testing with regard to their strategies and capabilities. The cohesiveness of the individual functions refers to the way their internal make-up offers a cohesive, integrated view of how the functions go about devising strategies and how they build and maintain capabilities.

Strategy cohesiveness concerns governance and resources and how these support the organizational scope of each function. Capability cohesiveness concerns the skills, tools, techniques, and methodologies used in software testing and development. The interdependencies that exist at the strategic and capabilities levels of each function of a corporate IT unit have to be in harmony (Thompson, 1967). This is because cohesiveness represents the level of congruence between the internal strategy or capability dimensions. In a sense it is logical to assume that cohesiveness of strategy and capabilities is a prerequisite for the other (between quadrant) flow components presented below.

Strategic Alignment

Strategic alignment of development and testing is the fit between development strategy and testing strategy (arrow 1 in Figure 1). Relative issues regarding the scope assigned to the development and testing functions, the governance of the functions, and their resource endowments are important to strategic fit. While these aspects do not have to be similar, they have to fit with each other, which implies that the interdependencies between them are harmonious and not dysfunctional (Porter & Millar, 1985). For example, each function's endowment of resources (financial, human, logistics, etc.) has to be adequate to cover interrelated responsibilities – responsibilities whose accomplishment both fulfills the function's own corporate role and enables its sister function to fulfill its role as well.

Capabilities Alignment

Capabilities alignment (arrow 2 in Figure 1) focuses on the comparative capabilities of development and testing at the operational/execution level and the operational linkage between their respective infrastructures, skills and processes. The interdependencies of these components have to be harmonious and enable each

function to achieve its goals. Dysfunctional relations between development and testing are often the result of misalignment of capabilities (Pettichord, 2000; Cohen et al., 2004; Ji et al., 2005; Zhang et al., 2008) rather than strategies. For example, the skills of testers have to be equal to or better than those of their developer counterparts for the testers to be able to provide adequate testing and quality assurance services. The methods, techniques, and tools used by testing and development also have to fit (work well) together for the two functions to efficiently and effectively do their jobs.

Execution Alignment

At times, an organizational unit that has a sound strategy and strong capabilities in place may not succeed because it is not able to execute its stated strategies. Execution alignment therefore refers to the ability of testing and development to operationalize or execute their strategies. Strategies conceptualized by upper echelon managers must be executed through technical, human, logistic, and procedural means. For example, if the tools or skills necessary to achieve strategic goals are not available in either function, then capabilities will be misaligned with strategy, and execution alignment will be low. Managers have to make sure that they have the proper resources – the tools, skills, and processes – to execute their stated strategy. Execution alignment of development (arrow 3a in Figure 1) refers to the ability of the development function to execute its stated strategy. Testing execution alignment (arrow 3b in Figure 1) is similar in that testing capabilities (competencies, tools, and methodologies) have to support the execution of stated testing strategies.

Research Methodology

To investigate the application of the alignment framework in a tactical setting, a longitudinal case study (Yin, 1984, 1994; Stake, 1995) was undertaken. Case study research is a proven means of providing in-depth and rich explanations about managerial and IT phenomena (Benbasat et al., 1987; Lee, 1989; Yin, 1994; Brown, 1999; Gallupe & Tan, 1999; Dwivedi & Kuljis, 2008). The case study approach allows researchers to delve deeply into phenomena and “flesh out” meanings and insights gained from a limited sample. It provides both “a theoretical account of the general features of a topic while simultaneously grounding the account in empirical observations or data” (Martin & Turner, 1986, p. 141). A single case study approach was used because DTA is primarily of concern in the context of the development of large, complex, real-time systems. When the context of a

phenomenon is of special interest, the case study approach allows for a better understanding of the complex relationships arising from the context (Orlikowski, 1993). This methodology is especially pertinent in our specific situation, in which distinct and separated development and testing groups reported to different parts of a complex IT organization. A careful assessment of alternative research methodologies suggested that quantitative approaches would be suboptimal given our research objectives, which involved interpretative construction of a contextual theoretical model. This process was informed by deductive analysis through review and adaption of applicable constructs from relevant past research. Finally, the resulting model was used in the case study analysis for interpreting what was occurring in the investigated environment. Figure 2 identifies the pertinent steps in our research.

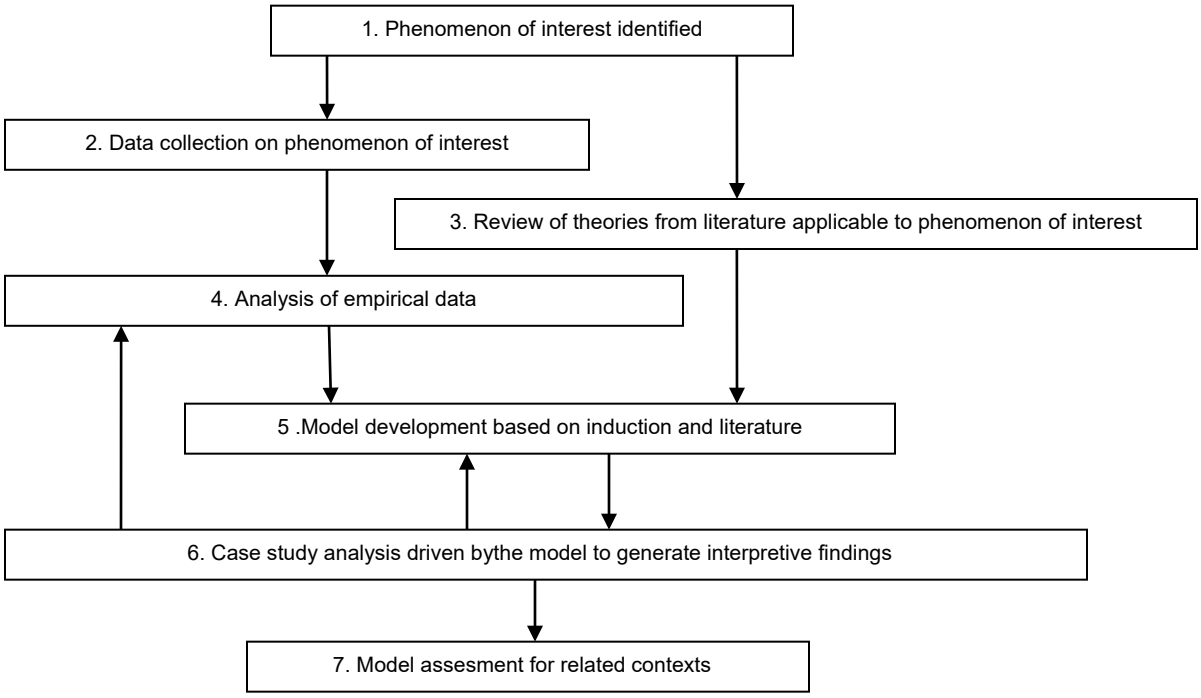

Figure 2: Research Methodology

The study took place over an eighteen-month period between September 2006 and April 2008 at a global Fortune 500 company that relies heavily on IT to conduct its business. This large organization has a distinct IT services group comprising more than 5000 developers and about 800 testers located at multiple locations

internationally. The corporate IT services group, which is led by a chief information officer (CIO), provides software, hardware, and system construction solutions for the information needs of the entire organization but customizes its solutions to the respective operating companies. Most of the required systems are developed in-house, making the software engineering function a paramount component of competitive strategy. To ensure the reliability and quality of software, the organization has, over the years grown a distinct testing sub-organization that functions separately but in coordination with the software development function. The development function was led by two vice presidents, focusing on different application systems. The testing group was led by a vice president who had the same rank and profile as the two development vice presidents, and all three of them reported up to a senior vice president who in turn reported up to the chief information officer. A formal software development life cycle was in place, but the vice presidents had flexibility and latitude in terms of strategic planning and executing tactical decisions. The development groups were separated not only by the applications they developed, but also by location. The testing unit was responsible for all application testing, but it too was split by location.

As researchers, we had initial interactions with the CIO and senior executives of testing and development because of their concerns about software engineering practices. This helped us identify the development-testing relationship as being the phenomenon of interest. We then had direct exposure to the organizational environment and had informal interaction with developers and testers at all levels of the organization. This data collection involved formal and informal sessions with executives and operating personnel from testing and development. Managers also gave us training sessions on the complex organizational processes and methodologies employed and on the technological platforms used by development and testing. These initial data collection steps allowed us to better understand the issues and, in combination with our review of the literature, directed us to alignment as a potential investigative lens for the phenomenon of interest. Subsequent iterations of theory development involving inductive and deductive cycles eventually led up to the model in Figure 1. To explore the application of the alignment lens and model, we then conducted ten group discussions of alignment issues such as resource endowment, scope of the function's activities, skills, processes, and so forth. The discussion groups were composed of testers (around 40% of the participants), developers (40%), and other stakeholders (requirements/marketing professionals, 20%). These discussions revealed that there were discontinuities between testers and developers at the strategic and capabilities levels. These interactions with key personnel were

followed up with further, in-depth, theory and model-driven interviews and observations designed to validate the theoretical model (please see Appendix 1 for a guiding protocol and an example of questions for participants).

Given the limited prior research, an interpretive approach was deemed appropriate for our exploratory study (Walsham, 1995). Building upon a central tenet of this approach, the socially constructed nature of knowledge (Boland, 1985; Hirschhiem & Klein, 1989), we undertook two distinct activities in our research. First, we presented our inductively derived initial theoretical model of DTA to about eighty researchers and practitioners in software development and testing in an effort to establish face validity and generate feedback as well as consensus on the value of our approach. Second, at a final stage we presented our findings back to the stakeholders in the case study organization to assess their perceptions of the value of our model and the pragmatic pertinence of our findings. To further enhance the socially constructed meaning of our model constructs, we also extended this process to obtaining the perceptions of testing and development professionals from seven other organizations. These presentations helped solidify our interpretations by providing additional feedback as to the applicability of our model while sensitizing us to the subjective and value-laden biases that may have shaped our analysis.

Secondary data was used to supplement insights from the multiple interviews that were conducted. The organization provided us with full access to its share-point portal where all internal documents were stored. In addition to notes and transcripts from interviews, our research database contains the organization's IT strategy documents, presentations used by managers and executives, descriptions of actual scenarios provided by individual testers and developers, formal training materials describing the organization's global development processes and internal control documents. These were all reviewed and analyzed at multiple stages of the research methodology using an iterative process (stages 4, 5, and 6 of our research methodology) as insights and interpretative findings emerged through deconstruction and understanding.

Findings and Discussion

Our case study analysis indicated significant disconnects in alignment between the development and testing units. On the whole we found more factors hindering than helping alignment. This was largely because our case study focused on an area that was a real business problem for our target organization. The alignment lens

we used permitted us to identify specific discontinuities, adaptations, and unanticipated trajectories pertaining to scope, governance, resources, competencies, and processes. We found several instances where formal and standardized processes where appropriated or adapted to obviate alignment difficulties between testers and developers. These findings relate to prior work by IT researchers, such as the interaction-oriented view of Leonard-Barton (1988), the adaptive structuration theory of DeSanctis and Poole (1994), and Robey's (1995) exposition of theories that explain the contradictory outcomes of IT.

A mid-level manager of the testing group had this to say about the perceptions of his IT subunit in the investigated organization: "*Both the senior IT level [managers] and all the development groups think that our work is systematic, structured and in compliance with what's on paper in the policy documents and the Powerpoints they see. The truth is that it is really chaotic, confusing, last-minute and hardly systematic. But we get it done somehow because we work hard and are creative explorers for defects. We are also masters at flexing around the structured processes of [name of the company omitted for anonymity*]."

Many of the participants pointed out that the scope of their activities was not well defined, especially in the areas of software testing. There were high expectations of both the development and testing units that were not ***matched*** by the authority over the software development process needed to meet those expectations. Participants complained about their lack of control over salient software development stages (e.g., requirements gathering), which were under the purview of business functions such as marketing. This state of affairs was exacerbated by ***incongruencies*** in the governance structures of the two units. Participants felt that the internal structures of their units were not efficient and effective enough in ***coordinating*** and organizing the internal operations of software development or in conducting efficient and effective ***coordination and communication*** with business stakeholders. The fact that the development unit had two VP-level executives, with different management styles, as well as a more rigid and formal control structure, while the testing unit was more participative, with a flatter and more flexible structure, led to internal conflicts and ***coordination and communication*** disconnects between developers and testers.

Another misalignment pertained to the relative resource endowments of the two units. While developers felt that the resources allocated to them were adequate for meeting their scope and responsibilities, testers felt that the resources available to them were woefully inadequate and did not ***match*** the scope and responsibilities assigned to them. Interestingly, developers did not perceive the testers as under-resourced. There was also a perception among testers that they were viewed as second-level employees, or as "failed developers," a view that

relegated them to the bottom of the resource allocation queue. This perceived lower status and shortage of resources led to low morale among testers and conflicts between testers and developers.

At the capabilities level, discontinuities stemmed from a gap in competencies and processes between testing and development. Respondents felt that developers had adequate skills and established processes for conducting software development. The competencies and processes used in testing did not ***match*** those used in development because of the relative difference in maturity.

The following sub-sections detail the findings of our case data analysis by structuring them on the salient structural and flow components of the development-testing alignment model.

Cohesiveness of Strategy and Capabilities

For each function (the columns of Figure 1) we assessed the cohesiveness of both strategy and capabilities. Appendix 2 presents our findings on the cohesiveness of each of the four structural components of our DTA model. We discuss the components that pertain to the strategy and capabilities of the development group before we discuss those for the testing group.

Cohesiveness of Development Strategy and Capabilities

The application of our model to the strategy component of the development group uncovered at least four areas of concern from an alignment perspective. First, we found inconsistency in the strategic scopes of the organization's development groups. Even though the overall IT organization had an integrated software development process, disparate development groups within the corporate IT unit had different interpretations of the scope of their responsibilities. This was a source of confusion and disagreement when these groups interacted with each other and with their testing counterparts. Another strategic incoherence stemmed from the fact that parts of the software development lifecycle were not under the control of the development group. For example, requirements gathering was generally a function of the marketing department, with developers having varying input in the process. This often led to misunderstandings about roles, the semantics of the requirements

gathered, and accountability for the quality of business requirements. These misunderstandings often trickled down to later stages of development, causing further misalignment. A development manager commented: "*The role of marketing in the [development process] is causing problems. They are not trained to write system requirements – let the developers do it. Development groups should have their own BA [Business Analyst] for requirements gathering using sophisticated methods like use cases. Marketing should not have a role in the [development process].*"

Third, we identified inconsistencies in the governance approaches of the development groups. One vice president of development emphasized informal and relational coordination mechanisms for management, while the other had a more formal, hierarchical approach. This difference in coordination style often led to breakdowns in coordination and communication between the team members of the two groups. These different styles of coordination were, to some extent, based on the personalities of the respective executives. Because of these breakdowns, development teams sometimes enacted group/team-based coordination (Van de Ven et al., 1976) without help from the formal hierarchical structure.

On the positive side, the development group as a whole was satisfied with the resources available to it. The perception was that budgeted resources were adequate to support the strategic scope set by upper management. Given the relative importance of software development in the overall strategy of the organization, the software development unit was given priority when budget allocations took place.

Application of the DTA model to evaluate the cohesiveness of the capabilities of the development group yielded five specific areas perceived to be sub-optimized. Many developers viewed the formalized SDLC-type process as outdated. In concordance with research by Giddens (1986) and findings by Soh (2000, 2003) and Boudreau & Robey (2005) we find that this formal, outdated SDLC process led to local adaptations of the process and the creation of undocumented workarounds and process exceptions that were not communicated and approved. While these local changes provided opportunities for process innovation and improvement, they were not properly disseminated to other interested parties in the organization. This created disconnects in the software development process that led to inefficiencies and delays. Second, a growing interest in the use of agile methods among new developers was leading to some confusion and conflict in team interactions, knowledge sharing, and the use of appropriate tools. Third, the use of automated tools in the development process was not aligned with the development methodologies being used. For example, the use of a requirements documentation

tool that was not integrated with development methodologies often led to excessive rewriting of code and misinterpretations about business rules and logic flows.

An area of positive influence was related to the software engineering skills available to the development group. The employees of the development group possessed good theoretical and practical experience in the area of software development and were able to complete their tasks with relative ease. However, there was a shortage of the skills required to use the new automated tools. The learning curve necessary for the acquisition of the required skills was creating a misalignment between the skills of the developers and the new software development tools.

Cohesiveness of Testing Strategy and Capabilities

Our analysis of the case study data showed a fairly positive picture of the cohesiveness of testing strategy. Generally, the testing group had proactive leadership that yielded well-specified strategy and planning. Testing's upper-level management was able to meet all the strategic requirements placed on it by their business and development counterparts. They were also able to translate these high-level strategies into actionable plans to be implemented by the testing unit and engender consensus on the scope of testing and its relationship to the function's governance. The scope provisioned by testing's upper management was well communicated and understood throughout the hierarchy of the testing unit. However, the resource base was incongruent with scope and governance. Generally, financial and human resources were tight for the testing group, given the strategic scope and responsibilities assigned to it. Location of testing teams in three separate geographical areas and the significant use of off-shore testing vendors were sources of some additional incoherence in the governance component of the organization's testing strategy.

In the area of cohesiveness of testing capabilities, the application of the DTA model uncovered five major areas for improvement. First, the variance in test planning and execution processes was causing instability within the testing group. Much of this instability was stemming from process factors, such as late arrival of code and poor allocation of testing resources to projects (based not on expertise, but on availability). Low skill levels in relation to architecture and process was a real concern and could be deemed the weakest link in cohesiveness. Many in the testing group had transferred in from development areas after performing poorly there. Highly competent personnel in the larger IT organization generally tried to avoid posting to the testing group because of its stressful work environment, largely caused by short turnaround times for testing code before release. The skills

required for the use of automated tools for testing were also generally not available in-house; instead it was the off-shore testing vendors who had these skills. One testing manager commented, "*We need to slow down absorbing all these new tools and methods. What about our skills to using them? Nobody thinks about that. Just because there is a new tool available doesn't mean we should be using it immediately. Process consistency is more important. Managers should think about this before jumping on the new gadget bandwagon to look good."*

Strategic Alignment of Development and Testing

Strategic development-testing alignment pertains to the degree of fit between the strategic focus of the development and testing groups on three distinct dimensions: scope, governance, and resources. Appendix 3 presents an overview of our findings in relation to the various aspects of this alignment.

Alignment of Scope

In terms of strategy, the scope of the two groups was well aligned on paper. This was because the organization had a well-formalized systems development life cycle that established clear responsibilities and processes for the role of each unit. A key aspect was the understanding that individual developers/programmers would complete unit testing for any module of code that they developed before it was passed on to the testing group. The testing group could then initiate integration and system testing on the basis of each module's internal consistency and correctness. However, in practice this demarcation between responsibilities for unit and integration testing was sometimes suboptimal from an alignment perspective. This was because the testing group had little influence over module specification in program design prior to the code being written. A standard practice of the development designers and coders was to decompose complex programs into a multitude of very small modules that minimized unit testing efforts for programmers but required testers to spend significant effort on testing them all together at a late stage. In essence, reducing the size of program modules shifted significant testing resource requirements away from unit testing by the development group to integration testing by the testing group, which had to test all the modules in an integrated manner. This indicated an incongruence in the goals of the development and testing groups, where development was focused on efficient delivery of working code, while testing was primarily concerned about verification and validation of the code in an integrated, requirements based, system related fashion (Elbert & Dumnke, 2007; Page, Johnston & Rollinson, 2009). During

our interviews, several participants suggested the idea of giving the testing group influence in the design process when module decomposition was performed to mitigate this scope misalignment, with its severe resource implications for the testing group.

Another issue pertaining to strategic scope alignment was organizational responsibility for developing and testing business requirements for new systems. The structured development life cycle of the organization tasked a separate marketing group (outside the corporate IT services organization) with the responsibility for developing systems requirements, in an effort to ensure that business requirements drove system development efforts. In a subset of cases, developers played an informal role in the requirements development process; however, their focus was seldom on validating requirements. Rather, they focused on assessing whether requirements were specific enough to allow designers and developers to begin coding.

The lack of a role for testers in testing requirements upfront often led to difficult situations later, when code had been written to specifications based on business requirements that had never been tested or validated. This finding supports the suggestion of Ji et al. (2005) who advocates the early engagement of testers in the software development process. While most of the executives we interviewed agreed that a role had to be found for testers in the early phases of requirements determination, there was insufficient strategic clarity around how this was to be achieved or why it had not been done. This lack of strategic clarity had significant implications for the adequacy of both testers' requirements validation training and their business understanding. The testing function felt that a strategic scope clarification was lacking that placed the responsibility for requirements testing on the testing group and ensured that appropriate resources were provided for these efforts. As a result, the development and testing groups remained strategically misaligned, with developers often arguing that their code was true to the specifications that they had been given, and testers arguing that a new system was not functional for meeting business needs.

A key alignment issue that arose during our investigation pertained to the systems development life cycle, which had been established long before the period of this study and prescribed standardized roles for both developers and testers. While everyone "had to follow" this life cycle process, there was no mechanism for addressing its inherent strategic role inconsistencies. In a sense, the life cycle was outdated, dating back to an era when systems were less complex and did not function in a global, web-based, real-time environment requiring much more flexible processes. One of the participants wondered:" *What's the point of having a complex and detailed methodology when no one is following it? Everyone is adapting and innovating around the [development*

process] because no one wants to challenge it. The people that imposed it many years ago are in senior management now and everybody is scared of criticizing the relic. Thankfully, there is flexibility informally for us to try new agile methods." While this flexibility allowed for reflexive local optimization and innovation (Soh et al., 2000, 2003; Boudreau & Robey, 2005) of the development process, the lack of tight communications between the development and testing units regarding these process changes meant that often developers and testers were following different, sometimes incongruent, software creation processes.

Another important dimension of strategic DTA is the comparative roles played by the two groups in shaping the future strategy and strategic orientation of the corporate IT unit. The developers, with their early role in the development cycle, had a stronger sense of being involved in shaping strategy and future priorities. The testing group, however, felt rather uninvolved in shaping future directions and priorities, given that their traditional role was to get going only when code was thrown their way. Historically, the testing group had not been led by a vice president–level executive; the current incumbent had been appointed only a few years prior to our investigation. Most testers and their managers felt that their role was not strategic within the corporate IT unit, and this was a source of both task and personal conflicts with developers. Related to the above were the relative levels to which developers and testers allowed input from other stakeholder groups (marketing, accounting, etc.) to shape their internal operational and strategic plans. Given the scanty interactions testers had with these external constituents, differences of opinion related to strategic scope were evident. Testers' interactions with others were largely limited to interactions with developers about technical issues related to correctness of code, while discourse with other stakeholders about longer-term policies, goals, and orientations was insubstantial and infrequent. This was a lesser issue with the development group, which enjoyed a higher organizational profile.

Alignment of Governance

A second key dimension of strategic development-testing alignment was the governance mechanisms used by the two groups. The development group was certainly numerically larger; the approximate overall ratio between developers and testers in the organization was more than 4 to 1. This meant that besides size, the development organization also had more levels of organizational hierarchy. This variance in levels suggested a possible misalignment in relation to strategic governance, with lower-level testers and testing managers often having to interact and deal with development professionals and managers who were at higher levels in the overall organizational hierarchy. Centralization is, in part, a function of the size of a group or organization (Ein-Dor &

Segev, 1982), and thus the testing group, with its much smaller number of individuals, was more centralized than the development group.

The formalization of planning also differed between the development group (which had more formalized planning processes because of its stability and size) and the testing group (which, in addition to its relatively smaller size, also had to contend with instability in work schedules). Testing was also done in response to code having been sent by developers. This stop/start reactive work behavior had not allowed the testers to formulate clear and formal plans and provisions regarding their activity.

Collaboration and coordination between and within the development and testing groups was largely project based, focusing on specific development modules. The two groups also differed on how within-group collaboration and coordination were effected. The development group had multiple project teams that competed in terms of efficiency as well as for a common pool of resources. The testing group had a more cooperative and collaborative environment because of its smaller size and lower organizational self-esteem. Testing thus had a more participative decision-making process than development. One of the testing directors commented: *"We are a more cohesive team because there are not many of us. The development teams are big and have high political issues. They are so disjointed. It's chaotic there. I am glad I moved to testing – we are a sharing, caring team."*

Another key aspect of governance pertained to how organizational strategy was provisioned, communicated, and executed within the two groups. Generally misalignment can result if one group has a bottom-up orientation while the other has a diametrically opposite, top-down orientation. Our research did not detect any variance between testers and developers on this aspect. Rather, the sense we obtained was that both groups accepted the largely top-down process.

Alignment of Resources

A third dimension of strategic alignment involves ascertaining the balance between the strategic resources possessed by developers and testers. If the two groups differ on strategic resource availability, organizational constraints, organizational clout, or in relation to their stated strategies for building competencies, DT alignment may suffer. Our analysis showed that the development group was comparatively better endowed in terms of knowledge, financial, and human resources. Its bigger size, longer track record, and the fact that many senior executives in the organization had systems development backgrounds also gave development more organizational clout, and better-established mechanisms for capturing and disseminating best practices. The

development group's more numerous, better-established formalized strategies and methods for building competencies was a source of misalignment, given the lack of such strategies in the testing group. For example, the testing group did not have a specified training and certification program for building up its knowledge base, while the development group had formalized and refined its training process over time. A positive influence on DTA was the fact that, as far as organizational constraints were concerned, both groups operated under the same assumptions.

The specificity of the IT group's strategic decisions pertaining to resource allocation and role specification was not in alignment with the mode of working prescribed by the outdated methodology used. Neither the development nor testing group had formalized and articulated strategic plans beyond budgetary allocations and goals. Since organizational resources were specified in accordance with the outdated life cycle methodology, while the actual work followed locally adapted versions of this life cycle, actual and budgeted resource requirements were mismatched. This lack of specificity of localized strategy meant that the development and testing units were often misaligned in terms of coordination and strategic intent.

In relation to resources, one of the interviewed testers said: *"Developers get everything – the most resources, positions, the best tools and their own schedules. Something must be done about having more sharing in the environments for development and testing. One group should not be so dominant."*

Capabilities Alignment of Development and Testing

This section investigates the relative levels of operational capabilities possessed by the two groups that impact their ability to execute their stated strategies. The dimensions of alignment of capabilities of development and testing are (1) the maturity of their work processes, (2) the quality of their skill sets, and (3) the sophistication of their tools, techniques, and methodologies. A summary of our findings is presented in Appendix 4.

We found that the two groups were significantly more misaligned in capabilities than in strategy. This was largely manifested in the process and skill aspects of capabilities and also in disconnects in architecture.

Alignment of Processes

The organizational and technical processes of the development and testing groups represent the first dimension of capabilities alignment. From our observations we saw that varying levels of process formalization

between development and testing led to disconnects in the communication and collaboration between the two groups. These disconnects were intricately tied to the differing operational methodologies of the two groups. The development group employed more formalized processes than the testing group, which had a more flexible mode of operation. This was partly a result of the different sizes of the two groups and partly a result of a difference in maturity. Developed and adopted in the mid 1970s, structured methods for systems development had been around for a longer time than structured methods for testing. The level of the two groups' process know-how was also different; the development group better understood the structured processes. This was a reflection of the state of industry best practices for software development. In regard to this, an interviewee observed: *"[The] testing group was brought together only a few years ago. It is just getting started on best practices. … Development is way better with this and has been doing it for so long. They even have knowledge management initiatives that we need to copy."*

Testing processes were generally less well developed and often characterized by ad hoc procedures and the uncoordinated use of "brute force" testing automation tools. This difference in the two groups' levels of process know-how and formalization was also reflected in variance in levels of process definitions. The development group had better-defined processes and specifications of processes. The testing group, on the other hand, had a less clear idea of the starting and end points of their processes and less precise mechanisms for completing each process step. In a sense, this could be a viewed as a positive characteristic of the testing group, as process flexibility and adaptability are important to testing, which requires creativity in finding bugs in code and innovative solutions when systems do not function as planned (Elbert & Dumnke, 2007; Page, et al., 2009). . Systems development, on the other hand, was primarily efficiency driven, focused on the need to meet clear specifications, a focus that clashed with the more creative, but less formalized and efficient, testing processes.

We also found different levels of stakeholder satisfaction with the outputs of the development and testing groups. The output of the testing group could be quantified more easily with metrics such as software bugs identified and corrections per lines of code. Constraints on the operational processes of the two groups were also different. Development processes could be more readily scheduled and budgeted. Testing processes, on the other hand, were characterized by late-stage stress, because code that had been written by developers often arrived late and close to software release deadlines. At other times, testers were frustrated by slack in work because they were waiting for code to arrive. The testing group had tried to overcome these wide fluctuations by structuring integrated load-planning cycles. While these were somewhat successful, they did not completely

eliminate the high peaks and low valleys characteristic of operational testing work. Testing still had to maintain excess capacity for the peak testing periods. Nearly everyone in testing agreed that more coordination and streamlining of development and testing processes was required to better align the capabilities of the two groups..

Alignment of Skills

The second dimension of capabilities DTA pertains to comparative skills available to the development and testing groups. Our analysis showed much larger gaps in alignment at the capabilities level than at the strategic level. We learnt that this was mainly because it was much easier to hire skilled developers in the software development marketplace than to hire skilled testers. Formal training in development was readily available both in academia and industry, but this was not the case for testing. The organization had recently recognized this source of misalignment and was planning to implement a customized training program for testers in an effort to reduce the skill gap. The significant gaps in the two groups' skills often led to interpersonal conflict between developers and testers. This analysis supports findings from prior literature (e.g. Pettichord, 2000; Cohen et al., 2004; Zhang et al., 2008) and provides a more granular explanation about the sources of this developer-tester conflict. In one interview, some developers characterized testers as "failed developers." Such perceptions of the capabilities of the testing group as against the "we are professionals" attitude of the development group were a key symptom of capabilities misalignment between developers and testers. One of the testing directors told us, "*Our testers are a loose bunch – they come from so many different areas – migrating in from not only development but also operations and marketing. … Skill levels in testing are all over the place as a result messing up our performance. The development group is more skilled and all have good technical background and good degrees. There are lots of them too."*

We further found that the development group had achieved a deeper level of knowledge about the functionality of the business applications being developed. The testing group had much less interaction with external stakeholders and focused more on technical errors, finding routines, and correction mechanisms. They had, over the years, developed less understanding of the organization's customers, products, processes, and marketplace needs. This meant that they were not as effective in validating the business relevance and appropriateness of the systems they were testing. Testing executives were cognizant of the fact that more testers had to be hired from business functions rather than from among those opting out of software development.

Another option that was being considered was hiring testers with both business knowledge and testing skills from outside the organization to boost testing capabilities.

Alignment of Architecture

The third and final dimension of capabilities DTA is comparative architecture. We compared the tools, techniques, and methodologies used for development and testing as well as the use of quality assurance methods, such as inspections and walkthroughs involving both testing and development personnel. The levels of activity standardization/coordination and technological constraints imposed on the development and testing groups were the other dimensions of architecture that were investigated.

We found that the overarching development life cycle of the IT organization was, at least in theory, amenable to integrating testing throughout the development process. In concordance with prior research (e.g. Ji et al., 2005) we found that such an early integration was considered beneficial and had a positive effect on alignment between the two units by bringing formal testing activities forward to the earlier requirements and design stages of development. Testing earlier in the life cycle was widely accepted as yielding significant time and cost savings and higher-quality products. Even so, the actual implementation of the overarching development life cycle was less than optimal and did not integrate testing into the software development process as well.

The use of quality assurance methods, such as walkthroughs and chief programmer team-based code inspections involving both developers and testers, also helped facilitate alignment. In our case study analysis, however, we found that, largely due to time and efficiency pressures, standard inspections and walkthroughs were often not implemented. Misalignment resulted, even though there were common intentions and instructions to undertake these procedures routinely.

In relation to the architectures for development and testing, we also probed the levels of standardization and flexibility of tools, techniques, and methodologies. We found that the development function had more flexibility for choosing tools and methods, because the development tools the organization had acquired over time were more mature than its testing tools, and because a larger selection of development tools could be secured from outside the organization. The testing group had less mature tools and techniques that were generally less proven with respect to architecture choices. We came to understand that this discrepancy reflected both the historical development of the organization (for example, there had been no executive-level leader for the testing group until recently) and the low number of effective tools and techniques that were viable options for acquisition

from outside vendors. This lack of alternatives in testing tools and methods imposed significant technological constraints on the testing group. The development group did not have to contend with this same level of architectural constraints.

Execution Alignment

The final flow component of DTA pertains to comparative execution capabilities: the two groups' ability to achieve their stated strategies and intent. The aspects of this component are generally similar to both strategic and capabilities development-testing alignment. Development and testing groups' variation in execution capability led to significant problems of misalignment. Largely, we found that the development group, with its longer history and greater maturity, could better execute its stated strategy as compared to the testing group. Many reasons underlay this finding, including resource availability, quality and skill of personnel, maturity of the formal organization, integrated tools available for development, and perceptions of the relative organizational importance of testers and developers. A symptom of such execution misalignment was regular interpersonal and task-related conflict between testers and developers (similar developer-tester conflicts have been identified by prior literature e.g. Pettichord, 2000; Cohen et al., 2004; Zhang et al., 2008). The pre-set release dates for software applications and the sequence of work wherein testing occurred after development was complete impeded the ability of the testing group to execute its strategy. Testers had to work under very stressful time constraints to complete their testing in time when code arrived late. Even though testing was internally aligned in strategic planning and capabilities, this reactive, last-minute "fire-fighting" orientation meant that the group often could not execute its strategies in a coherent and efficient manner, given that it did not want to be left "holding the ball" when release dates for software were not met. There was recognition in the organization that more had to be done to move testing activities to the early parts of the development life cycle if this misalignment was to be overcome.

Another finding was the need for coupled and integrated automated development and testing platforms to ensure that testing could begin earlier and that the start-up time for testing activities would be minimized. Another impediment holding back the ability of the testing group to execute its strategy was the fact that it did not have control over the operational production platforms on which the business applications actually ran after being released. It was commonly felt that unless the testing group was given some level of responsibility for these platforms, testing would remain impeded in its ability to do a good job validating code. Better documentation requirements for development and more tester-friendly development practices were also identified as ways to

facilitate the work of the testers and allow them to execute their strategies. Many interviewees suggested that a revision of the systems development life cycle in the organization to better integrate development and testing work, and a bilateral job rotation program between development and testing, were other ways the strategic execution capability of the testing group could be improved. The organization was also considering a certification program in testing for both testers and developers to better align the comparative execution capabilities of the two groups.

Conclusion and Directions for Future Research

Our first research goal was to explore the viability of applying alignment concepts at a less strategic, micro level within a corporate IT unit and, specifically, to development and testing groups. On the whole, we found support for this as the alignment model of Figure 1 could intuitively be applied to the case of the two IT subunits we investigated to yield valuable insights. Unlike many other strategic frameworks, alignment concepts have potential for application within the context of the corporate IT unit. We presented our model and findings to executives and professionals from both the testing and development groups in several iterative cycles that helped us sharpen the findings and validate our interpretations. Without exception, all stakeholders highly rated our formulation of the development-testing relationship in terms of our DTA model. They appreciated how this approach helped them take a strategic perspective while uncovering specific misalignment dimensions that could serve as the basis for managerial intervention. The diverse segments of our alignment model that distinguished strategies from capabilities and our explicit focus on execution for the two subunits all helped to make our model viable and valuable. A key dimension of our model, the focus on the ***cohesiveness*** of strategies and capabilities, was identified as its most valuable part. This dimension has not been considered in prior research (e.g. Henderson & Venkatraman, 1993; Reich & Benbasat, 2000; Sabherwal et al., 2001; Luftman & Kempiah, 2007; Preston & Karahanna, 2009). Our granular view of alignment also permitted the identification of some of the causes of conflict between developers and testers. Literature has identified such conflicts before (Pettichord, 2000; Cohen et al., 2004; Zhang et al., 2008), but our alignment lens provides a theoretically driven framework for uncovering additional sources of such conflict that are contextualized to both strategic and tactical realms.

Our second goal was to use alignment as a theoretical lens to explore context-driven organic phenomena and to generate insights/understanding about discontinuities and unanticipated trajectories in IT management. We found that the alignment lens yielded a rich set of interpretations and explanations. It helped isolate alignment and misalignment issues, and provided strong clues as to the sources and history of the issues. Comments made by the stakeholders indicated that our model and findings enabled them to better understand the intricacies of the long-standing disruptive relationship between development and testing. Many suggested that our approach would yield better solutions than existing approaches involving reactive conflict resolution, mediation, and established procedures for escalating conflicts to higher organizational levels. Linking back to the prior literature on strategic alignment (Venkatraman, 1989; Henderson & Venkatraman, 1993), we found that a key difference in our study was that we utilized various conceptualizations of fit in alignment (such as matching, coordination, covariation, communication, and congruence) in our case study analysis. Unlike prior cross-sectional empirical studies of strategic alignment, these yielded a deeper set of rich explanations and interpretations, as evidenced in our findings.

Our third goal was to empirically link alignment to essential management challenges faced by technology organizations, such as management style, process and organizational issues, skill discontinuities, and unintended appropriations of technology, etc., that facilitate managerial understanding and responses. Our analysis conveys additional support to findings from prior literature on appropriations of technology (e.g. Soh et al., 2000; 2003; Boudreau & Robey, 2005) by showing how individuals and groups in testing and development appropriate IS technology and processes and adapt them to the local context. We further show that these informal appropriations, if improperly communicated and aligned with other groups, can be disruptive to the overall software engineering process. Our model, analysis and findings proved to be readily assimilated by all levels of development and testing stakeholders further supporting our interpretation of the contextual findings. All three vice presidents of both groups accepted our model as being pertinent and relevant. They found our evaluation and data analysis to converge with their perceptions of the status quo and rated the new strategic perspective provided by our model as superior to prior assessments and approaches.

To further strengthen the basis of these findings and ascertain the study's applicability, we shared our DTA theoretical lens and findings with representatives of other pertinent organizations and institutions. As our organizational context was not selected through systematic sampling, our findings are not generalizable to wider contexts. This final methodological step helps ensure that our interpretations are meaningful to others seeking a

starting-point, frame of reference for understanding the interactions between developers and testing in software engineering. This final step yielded positive responses and feedback as to the pragmatic value of our alignment model and analysis of the internal dynamics between IT subunits in terms of alignment. These discussions also sensitized us to aspects of our context that were driving certain unintended consequences, such as the differing managerial styles of the vice presidents. The discussions also provided us with deeper insights into the limits and biases of our own interpretation. The alignment approach used by our study can serve as a viable managerial mechanism for others seeking to explore alignment between two IT subunits, and can provide the basis for both managerial understanding and action planning.

Directions for future research

A key contribution of our research is the conceptualization of alignment as a new theoretical lens for understanding the often difficult relationship between the software development and testing functions. Our results suggest that this approach could also be pertinent for the investigation of other internal relationships between distinct groups within corporate IT units. This could be the focus of future research, given that corporate IT units comprise a diverse set of constituencies with multiple and overlapping interdependencies that have to be aligned for success. Theoretical focus on the alignment of strategy, capabilities, and execution can help shift the focus away from the short-term “fire fighting” and “problem escalation” orientations common in corporate IT units to a longer-term strategic orientation. This could also be the basis for future research in this area, given that senior IT executives constantly struggle to maintain the strategic focus of their units in the face of day-to-day demands for IT services.

Another useful direction for future research could be development of internally valid and generalizable metrics for the specific components and dimensions that our study has identified. A useful way of doing this could be to distinguish between managerial and technical software engineering alignment measures and metrics. Given the close interplay between such constructs and measurements, future research work can help validate our current findings as well as identify limitations of the alignment approach we are espousing.

Given the use of a single case in our research study, it is our belief that more research can be useful in fostering more robust theoretical and practical ideas about organizing and aligning software testing to improve the efficiency and effectiveness of software engineering. We also recognize the possible limitations of reliance on

subjective perspectives in data collection and analysis. The use of quantitative methodologies to provide different views of the phenomenon may also be useful as an area of future research.

Another key area for future research is the investigation of the impacts of developer-tester alignment. The focus here needs to be on questions such as the following: (1) Does such alignment reduce software development time? (2) Does such alignment help developers improve their coding abilities and methods on the basis of feedback from testers? (3) Does such alignment reduce software development costs?, and (4) Does such alignment improve the quality of the software developed?

A final hope is that our research will motivate future investigations of alignment between subgroups within corporate IT especially when such tactical alignment is missing.

References

AVISON D, JONES J, POWEL P, and WILSON D (2004) Using and Validating the Strategic Alignment Model. *Journal of Strategic Information Systems* 13, 223-246

BENBASAT I, GOLDSTEIN D, and MICHAEL M (1987) The Case Research Strategy in Studies of Information Systems. *Management Information Systems Quarterly* 11(3), 369-386

BLACK R (2002). *Managing the Testing Process*. (2nd ed.) John Wiley & Sons, Inc. New York, NY, USA.

BOAR BH (1984) *Application Prototyping: a Requirements Definition Strategy for the 80s*. John Wiley & Sons, Inc. New York, NY, USA

BOUDREAU M-C and ROBEY D (2005) Enacting Integrated Information Technology: a Human Agency Perspective. *Organization Science* 16(1), 3–18

BRANCHEAU JC, JANZ BD and WETHERBE JC (1996) Key Issues in Information Systems Management: 1994-1995 SIM Delphi results. *Management Information Systems Quarterly* 20(2), 225-242

BROWN CV (1998) Advantage 2000 at Owens Corning, In *Managing Information Technology: What Managers Need to Know* in BROWN EW, DEHAYES CV, HOFFER DW, J A and PERKINS WC (1999), Eds, pp.640-658, Prentice Hall, Upper Saddle River, NJ

COHEN C F, BIRKIN SJ, GARFIELD MJ and WEBB HW (2004) Management Conflict in Software Testing. *Communications of the ACM* 47(1), 76-81

CHAN Y, HUFF S, BARCLAY D, and COPELAND D (1997) Business Strategic Orientation, Information Systems Strategic Orientation, and Strategic Alignment. *Information Systems Research* 8(2), 125-150

DESANCTIS G and POOLE MS (1994) Capturing the Complexity in Advanced Technology Use: Adaptive Structuration Theory. *Organization Science* 5(2), 121-147

DWIVEDI YD and KULJIS J (2008) Profile of IS research published in the European Journal of Information Systems. *European Journal of Information Systems* 17(6), 678-693

EIN-DOR, P and SEGEV E (1982) Organizational Context and MIS Structure: Some Empirical Evidence. *MIS Quarterly* 6(3), 55-72

GALLUPE RB and TAN FB (1999) A Research Manifesto for Global Information Management. *Journal of Global Information Management* 7(3), 5-18

HENDERSON J and VENKATRAMAN N (1993) Strategic Alignment: Leveraging Information Technology for Transforming Organizations. *IBM Systems Journal* 32(1), pp. 472-484

HIRSCHHEIM R. and KLEIN HK (1989) Four Paradigms of Information Systems Development, *Communications of the ACM* 32(10), pp. 1199-1216

HUSSIN H, KING M and CRAGG P (2002) IT Alignment in Small Firms. *European Journal of Information Systems* 11(2), 108-127

IEEE Std. 1028-1997, IEEE Standard for Software Reviews, clause 38.

JI Y, MOOKERJEE V and SETHI S (2005) Optimal Software Development: A Control Theoretic Approach. *Information Systems Research* 16(3), 292-306

LEONARD-BARTON D. (1988) Implementation as Mutual Adaption of Technology and Organization, *Research Policy* 17(5), pp. 251-267

LESCA N, CARON-FASAN ML (2008). Strategic Scanning Project Failure and Abandonment Factors: Lessons Learned. *European Journal of Information Systems* 17(4), 371-386

LUFTMAN J, PAPP R and BRIER T (1999) Enablers and Inhibitors of Business-IT Alignment. *Communications of the Association for Information Systems* 1(3), 1-32

LUFTMAN J and KEMPAIAH R (2007) An Update on Business-IT Alignment: A Line Has Been Drawn. *MIS Quarterly Executive* 6(3), 165-177

MALONE TW and CROWSTON K (1991) Toward an Interdisciplinary Theory of Coordination (Working Paper No. 120), MIT Centre for Coordination Science. Boston, MA

MARTIN PY and TURNER BA (1986) Grounded Theory and Organizational Research. *The Journal of Applied Behavioral Science* 22(2),141–157

MCKEEN JD and SMITH H (2003) *Making IT Happen: Critical Issues in IT Management*. Wiley, Chichester, Hoboken, NJ

MILLER D. (2008) Keynote Address, In *International Workshop on Advances and Innovations in Systems Testing*, FedEx Institute of Technology, Memphis, May 6, 2008

ORLIKOWSKI WJ (1993) CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development. *MIS Quarterly* 17(3), 309–340

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

PORTER ME and MILLAR V (1985) How Information Gives You Competitive Advantage. *Harvard Business Review* 63(4), 149-160

PRESTON DS and KARAHANNA E (2009) Antecedents of IS Strategic Alignment: A Nomological Network. *Information Systems Research* 20(2), 159-179

REICH BH and BENBASAT I (1996) Measuring Linkage between Business and Information Technology Objectives. *MIS Quarterly* 20(1), 55-81

REICH BH and BENBASAT I (2000) Factors that Influence the Social Dimension of Alignment Between Business and Information Technology Objectives. *MIS Quarterly* 24(1), 81-113

REICH BH, and NELSON KM (2003) In their own Words, CIO Visions about the Future of In-house IT Organizations. *The DATA BASE for Advances in Information Systems* 34(4), 28-44

ROBEY D. (1995) Theories that Explain Contradiction: Accounting for the Contradictory Orgainzational Consequences of Information Technology, *Proceedings of the Sixteenth International Conference on Information Systems*, Amsterdam, pp. 55-63

SABHERWAL R, HIRSCHHEIM R and GOLES T (2001) The Dynamics of Alignment: Insights from a Punctuated Equilibrium Model. *Organization Science* 12(2), 179-197

SOH C, KIEN KS, BOH WF and TANG M (2003) Misalignments in ERP Implementation: A Dialectic Perspective, *International Journal of Human–Computer Interaction* 16(1), 81–100

SOH C, KIEN KS and TAY-YAP J (2000) Cultural Fits and Misfits: is ERP a Universal Solution? *Communications of the ACM* 43(4), 47–51

STAKE RE (1995) *The Art of Case Study Research*. Sage Publishing, Thousand Oaks, CA

TEO TSH and KING WR (1996) Assessing the Impact of Integrating Business Planning and IS Planning. *Information and Management* 30(6), 309 – 321

TALLON PP, KRAEMER KL and GURBAXANI V (2000) Executives' Perceptions of the Business Value of Information Technology: A Process-Oriented Approach. *Journal of Management Information Systems* 16(4), 145-173

THOMPSON JD (1967) *Organizations in Action*. McGraw-Hill, St Louis, MO

VAGOUN T and HEVNER AR (1997) Feasible Input Domain Partitioning in Software Testing: RCS case Study. *Annals of Software Engineering* 4, 159-170

VAN DE VEN AH, DELBECQ Al and KOENIG K (1976) Determinants of Coordination Modes within Organizations. *American Sociological Review* 41, pp 322-228

VENKATRAMAN N (1989) The Concept of Fit in Strategy Research. *Academy of Management Review* 14(3), 423-444

WATSON RT, KELLY GG, GALLIERS RD and BRANCHEAU J (1997) Key Issues in Information Systems Management: an International Perspective. *Journal of Management Information Systems* 13(9), 91-116

YIN RK (1984) *Case Study Research: Design and Methods*. Sage Publications, Newbury Park, CA

ZHANG X, DHALIWAL J, GILLENSON M and MOELLER G (2008) Sources of Conflict between Developers and Testers in Software Development**.** *Proceedings of 14th Americas Conference on Information Systems (AMCIS 2008),* Toronto, Ontario, Canada, August 14-17

Appendix 1: Model-Driven Guiding Protocol for Interviews and Group Discussions

Cohesiveness of Development Strategy	Scope and Governance	Does the governance structure of the development unit ***match*** its mission? How well does the hierarchical makeup of the development unit ***fit*** with the scope of its activities? Is the governance structure of the development unit flexible in terms of ***co-variation*** in facilitating adequate control as the scope of development varies on projects and over time? Are there proper ***coordination/communication*** channels between the development unit's stakeholders and its internal hierarchy?
	Governance and Resources	Is the governance structure of the development unit ***appropriate*** for the purposes of securing resources needed to ***match*** its mission? Is the internal structure of the development unit ***congruent*** with the resource allocation process in the organization? Is the governance structure of the development unit appropriate for ***coordinating*** the internal allocation of resources?
	Scope and Resources	Do the resources allocated to the development unit ***match*** the need to cover the scope of its activities? Do the resources allocated for development ***co-vary*** with the development scope that is assigned on projects? Is there a ***shared understanding*** at various levels of the organization about the appropriate resources required for accomplishing development goals?
Cohesiveness of Development Capabilities	Processes and Architecture	Explain how well the development architecture (tools, techniques and methodologies) ***match*** development processes. Do the tools and techniques used for development ***coordinate*** well with the specified development processes? Are development processes flexible in adapting and ***co-varying*** as architectural tools, techniques and methodologies change?
	Architecture and Skills	Do the development unit's skill sets ***match*** the tools and techniques it uses? Is there a good ***fit*** between the skills of developers and the types of tools and techniques used? Do development skill sets change and ***co-vary*** with changes in tools, techniques and methodologies employed?
	Skills and Processes	Is there a ***gap*** between the competencies of developers and the processes employed for software development? Is there a ***common understanding*** about the skills and processes required for successful software development? Is the ***mutual interdependence*** between development processes and skills available considered in development decisions?
Cohesiveness of Testing Strategy	Scope and Governance	Does the governance structure of the testing unit ***match*** its mission? How well does the hierarchical makeup of the testing unit ***fit*** with the scope of its activities? Is the governance structure of the testing unit flexible in terms of ***co-variation*** in facilitating adequate control as the scope of testing varies on projects and over time? Are there proper ***coordination/communication*** channels between the testing unit's stakeholders and its internal hierarchy?

	Governance and Resources	Is the governance structure of the testing unit ***appropriate*** for the purposes of securing resources needed to ***match*** its mission? Is the internal structure of the testing unit ***congruent*** with the resource allocation process in the organization? Is the governance structure of the testing unit appropriate for ***coordinating*** the internal allocation of resources?
	Scope and Resources	Do the resources allocated to the testing unit ***match*** the need to cover the scope of its activities? Do the resources allocated for testing ***co-vary*** with the testing scope that is assigned on projects? Is there a ***shared understanding*** at various levels of the organization about the appropriate resources required for accomplishing testing goals?
Cohesiveness of Testing Capabilities	Processes and Architecture	Explain how well the testing architecture (tools, techniques and methodologies) ***match*** testing processes. Do the tools and techniques used for testing ***coordinate*** well with the specified testing processes? Are testing processes flexible in adapting and ***co-varying*** as architectural tools, techniques and methodologies change?
	Architecture and Skills	Do the testing unit's skill sets ***match*** the tools and techniques it uses? Is there a good ***fit*** between the skills of testers and the types of tools and techniques used? Do testing skill sets change and ***co-vary*** with changes in tools, techniques and methodologies employed?
	Skills and Processes	Is there a ***gap*** between the competencies of testers and the processes employed for software testing? Is there a ***common understanding*** about the skills and processes required for successful software testing? Is the ***mutual interdependence*** between testing processes and skills available considered in testing decisions?
Strategic Alignment between testing and development	Scope	Is there a ***match*** between the scope of the testing unit and that of the development unit? The testing and development units ***coordinate*** well in relation to changes in goals and scope of projects? Efforts are made to develop ***shared understanding*** about the relative scope of development and testing on projects?
	Governance	Is there a ***match*** between the governance structures of the development and testing units? Is there ***congruence*** between the organizational structures of the development and testing units? Do the governance structures of testing and development ***co-vary*** to changes in projects and the external environment? Is there adequate ***coordination*** between all levels of the testing and development units?
	Resources	Are the relative resources allocated to testing and development reflective of the ***interdependence*** between them? Do the resources allocated for testing and development ***co-vary*** with changes in strategic scope and risk in software deployment? Is there ***shared understanding*** between testers and developers about resource availability and need?
Capabilities Alignment between testing and development	Processes	Is there a ***match*** between the process capabilities employed by the testing and development units? Do the process capabilities of testing and development ***co-vary*** to meet particular project needs? Is there good ***communication and coordination*** between development and testing about software deployment processes?

	Architecture	Do the capabilities of the tools, techniques and methodologies used for development ***match*** those used for testing? Is there recognition that the ***inter-dependence*** between the architectural capabilities of the development and testing units impacts their relative performance? Is there adequate ***coordination and communication*** between testers and developers in relation to the use of tools, techniques and methodologies?
	Skills	Is there a ***match*** between the relative skills and competencies of the testing and development units? Do testers and developers improve their skills and competencies to keep up with ***variations*** in technologies and methods? Is there effective ***communication and coordination*** about training plans for testers and developers?

Execution Alignment of Development	Is there a ***match*** between the processes, skills and architectural capabilities of the development unit in terms of ability to execute its stated strategies? Is there enabling ***co-variation*** between changes in development strategies and capabilities that impact the execution ability of the development unit? Are there proper internal ***communication and coordination*** mechanisms necessary for the execution of development strategies and plans given its capabilities?
Execution Alignment of Testing	Is there a ***match*** between the processes, skills and architectural capabilities of the testing unit in terms of ability to execute its stated strategies? Is there enabling ***co-variation*** between changes in testing strategies and capabilities that impact the execution ability of the testing unit? Are there proper internal ***communication and coordination*** mechanisms necessary for the execution of testing strategies and plans given its capabilities?
Relative Alignment of Ability to Execute	Are there differences in the relative ability of the testing and development units in terms of their ability to execute their respective strategies?

Appendix 2: Findings for Cohesiveness of Strategy and Capabilities

Cohesiveness	Findings
Development Strategy	Inconsistent scope between development groups within the organization.
	Key software development steps were outside of the scope of the development group (requirements gathering was primarily function of the marketing group).
	Inconsistent governance structure between the distinct development groups within the organization – ambiguous hierarchical structure between the various executive-level managers of development groups.
	There were two vice presidents of the development group, with different governance approaches (personality based).
	Development personnel were generally satisfied with their resource base, especially in relation to the strategic scope of the group.
Development Capabilities	Formalized SDLC-type processes viewed as outdated by most developers, resulting in workarounds and process exceptions.
	Explosion in the use of agile methodologies was leading to confusion about processes and required interactions.
	Automation tools were not integrated with the development methodology.
	Developer skills were misaligned with requirements for new automated tools.
	Developers had the necessary software engineering skills required to enable their processes.
	The lag in the learning curve arising from the introduction of new development tools created a misalignment between the skills of the developers and the tools in use during development.

Testing Strategy	The resource base was incongruent with the ambitious scope and expectations (loads) specified in testing strategy.
	Geographical dispersion of testing teams was a source of misalignment.
	Proactive leadership of the testing group yielded well-specified strategy and planning.
Testing Capabilities	Variance in test planning and execution processes caused instability within the testing group.
	Skills in the use of new, automated testing tools were inadequate.
	Use of multiple vendors for outsourced testing led to inconsistencies in techniques and processes.
	The new tools, techniques, and methodologies were exacerbating the variance in processes and skill requirements.
	Basic skill-sets in testing were the weakest link among the cohesiveness components.

Appendix 3: Findings for Strategic DT Alignment

Strategic DTA	Finding
Scope of Development and Testing	Well-defined scope regarding unit vs. integration testing on paper but not in practice.
	Testers and developers minimally involved in requirements phase.
	Development and testing strategies driven by overly high-level SDLC that provided little specificity for resource allocation and role definition in localized decision making.
	Development groups had a stronger role in longer-term strategic decisions, while testers perceived their role as being of lesser strategic significance.
	Testers had little interaction with external stakeholders as compared to developers.
	Both groups had a top-down strategy provisioning process.
Governance of Development and Testing	Larger, more centralized development organization as compared to smaller and more decentralized testing organization.
	Testing organization had fewer hierarchical layers than development.
	Development organization had more formalized planning processes because of stability as compared to reactive orientation of testing work.
	Smaller testing group had more collaborative environment. Larger development group had greater coordination issues.
	Development group had a more diverse decision-making structure as it had two vice presidents as opposed to only one for testing
Resources of Development and Testing	Development group were better endowed with resources than the testing group.
	Development groups had more sophisticated best-practices
	Both groups were faced with common organizational constraints.
	Testing group had lower organizational profile and influence than the development group.
	Development groups had more experience with strategies for building competencies. Testing had weak training and certification programs.

Appendix 4: Findings for Capabilities DT Alignment

Capabilities DTA	Finding
Processes for Development and Testing	The larger and more mature development group employed more formalized processes than the flexible testing group.
	Development has a better understanding of its process due to the relative maturity of the software development.

	Development group has a better grasp of the starting and end points of processes as well as more precise implementation mechanisms. Testing has looser definitions because process start and end are driven by defect histories and complexities.
	Testing output is more easily quantifiable and provides a clearer assessment than development output.
	Development processes are better planned and provisioned for, while testing processes are reactive and subject to uncontrollable events that lead to fluctuating workloads.
	More mature and prominent development function benefits from more extensive interaction with external stakeholders.
Skills for Development and Testing	A comparative abundance of skilled developers in the organization was observed. Testers tend to migrate in from other units and do not have consistency in skills-level or effectiveness.
	The training programs of the development group are more mature.
	The development group has achieved a deeper level of understanding of the business than the testing group.
Architecture for Development and Testing	Both groups used complementary methodologies in their activities.
	The development group, due to its size and longer tenure, employed a more standardized architecture than the smaller testing group which had more coordination issues in terms of tools and techniques.
	Although the QA methods were aligned on paper, in practice the testing and development groups did not employ common QA methods.
	Testing tools employed were less developed than development tools. The testing group had more technological constraints as a result.