Managing Inherent Conflicts in Agile Distributed Development: Evidence from Product Development

Ashay Saxena
Indian Institute of Management Bangalore, ashay.saxena@iimb.ernet.in

Shankar Venkatagiri
Indian Institute of Management Bangalore, shankar@iimb.ernet.in

Rajendra Bandi
Indian Institute of Management Bangalore, rbandi@IIMB.ERNET.IN

Managing Inherent Conflicts in Agile Distributed Development: Evidence from Product Development

Ashay Saxena
Information Systems
Indian Institute of Management Bangalore
Bangalore, India
Email: ashay.saxena@iimb.ernet.in

Shankar Venkatagiri
Decision Sciences & Information Systems
Indian Institute of Management Bangalore
Bangalore, India
Email: shankar@iimb.ernet.in

Rajendra K Bandi
Decision Sciences & Information Systems
Indian Institute of Management Bangalore
Bangalore, India
Email: rbandi@iimb.ernet.in

Abstract

Increasingly, software is being developed following agile approaches in a distributed setup. An agile setting is typically characterized by *flexibility*, in order to accommodate changing customer demands for continuous delivery of business value. A distributed setting brings about multiple demands for *stability*, in terms of a push for clear specification of requirements and design, and a big picture product definition. Therefore, implementing agile distributed development projects results in an inherent conflict that must be reconciled. We conducted two case studies of such projects in the domain of product development to examine the nature of conflict as well as the mitigating mechanisms followed by the software teams. Our findings reveal that the domain of engagement drives the need for flexibility, and the specific distributed team configuration drives the demand for stability. Furthermore, the software teams achieve a balance between them through the project context, which is characterized by an interaction of performance related and social elements.

Keywords

Agile Distributed Development, Flexibility-Stability Conflict, Ambidexterity, Case Study

1 Introduction

Up until the 1980s, organizations developed software using a plan-driven approach, with a big up-front requirements exercise, followed by well-demarcated design, coding, testing and rollout stages. This waterfall model had several drawbacks; chief among them was a reluctance to admit and implement changes, which frustrated project sponsors. At the turn of the century, organizations began to seek and embrace 'better ways of developing software' (Agile Alliance 2001). Agile as a philosophy was expressly conceived to address the twin demands of accommodating volatile requirements from the customer, while delivering working software in quick increments. A strong emphasis on team collaboration necessitated extensive face-to-face communication; initially, agile practices assumed the context of a single, collocated team. Over time, internet bandwidth became ubiquitous and inexpensive, which made way for powerful video conferencing and group collaboration tools; the restriction of collocation for agile software development has since been vastly relaxed.

Over the last two decades, organizations have begun to execute their projects using teams that are located at geographically dispersed software development centres. This move has been triggered by contingencies such as faster time-to-market, diverse competencies, and prevailing wage differences for resources across the globe. The impact of globalizing development is that at an architectural level, products must now be designed with a higher degree of modularity. This decouples software into independent stages, so they can be designed, developed and tested at different sites across the globe. Alternately, the software can be decoupled into features or components, which are developed at multiple sites with a fair degree of autonomy. Keeping pace with these developments, agile efforts have become globally distributed, with a mission to deliver high quality software incrementally, and in multiple time-boxed iterations (Shrivastava & Date 2010).

Agile Distributed Development (ADD) is a model in which projects are implemented by following an agile methodology, with teams that are located across multiple geographies. ADD engagements might involve building an "off-the-shelf" product for a wide market, or servicing the needs of a specific client with custom developed software. The "services" model has received considerable attention in literature (e.g., Ramesh et al. 2006; 2012). Despite increasing attention from the software community, the literature on ADD product engagement models is rather sparse. Such efforts involve building a new platform, or incorporating features into an existing one; this demands innovative thinking from teams that are distributed across geographies. It is commonly understood that the requirements of the end customer are indirectly represented by a proxy group; the software producer bears complete responsibility for any risks that arise (Fogelström et al. 2010). These characteristics have direct implications on ADD practice, beyond the contingencies imposed due to the distributedness.

Existing literature suggests that executing agile projects in distributed setting presents an inherent conflict (Ramesh et al. 2006, 2012; Lee et al. 2006). The distributed setting is characterized by spatial, temporal, and configurational differences (O'Leary & Cummings 2007). Generally, dispersed team members develop weak ties (Wong & Burton, 2000); they rely on ICT-mediated interactions to coordinate their tasks. As a result, software teams operating in a distributed setup prefer to work towards delivering what they know has been finalized upfront. On the other hand, agile teams must work with frequent updates to the project plan that are dictated by changing requirements (Agile Alliance 2001). They actively seek regular feedback from customers on their work, and in turn provide rapid business value to them through short iterative cycles.

Whereas distributed teams seek *stability* to predictably meet the project objectives, agile teams require *flexibility* to continuously update the software artefact and deliver functionality. Naturally, there is a conflict between these demands. Dealing with such a conflict is not unique to the ADD setting. Researchers (e.g., Gibson & Birkinshaw, 2004) have examined other domains that involve demands of a similar nature for alignment and adaptability, across an entire business unit. However, the contingencies experienced by software teams operating in a product development setup remains to be examined. Hence, our study is driven by the research question:

> **RQ1**: *What is the exact nature of the flexibility-stability conflicts that arise in agile distributed product development settings?*

The capabillity of organizational ambidexterity provides guidance in balancing the trade-off. Ramesh *et al.* (2006; 2012) point out the infeasibility of a structurally ambidextrous response (Tushman and O'Reilly 1996), which involves splitting the organization into collocated agile and globally distributed divisions. Given that the two aspects are inseparable for ADD, they suggest developing contextually ambidextrous responses (Gibson & Birkinshaw 2004), to handle the problem in whole. For this, the teams must innovatively synthesize responses that mix formal and informal elements. For example, individuals in a software team could have a precise description of their roles and responsibilities, but have the freedom to reach out to their counterparts across sites for any clarifications. Such responses need to be further examined across ADD engagements to bring more clarity on the mechanisms followed by software teams to manage the trade-off. Keeping our focus on product-oriented ADD engagements, we state another research question:

> **RQ2**: *What mechanisms do software teams devise to achieve a balance between flexibility-stability in agile distributed product development settings?*

To address these questions, we have followed an exploratory case study approach. We analyze two ADD project teams that are engaged in developing products concerning (1) a system-on-chip solution, and (2) a healthcare cloud platform, respectively. The contribution of this work lies in providing insights, in the form of nuanced clarity on the notion of flexibility-stability conflict and how contextual ambidexterity forms the basis for devising enablers to mitigate such forces.

The remainder of the paper is organized as follows. The next section presents a review of the literature on agile distributed development, software product development engagement, and a theoretical background on the notion of ambidexterity. Section 3 describes the research design. Section 4 presents the findings from our case studies that illustrate the inherent conflict faced in the setting, as well as the enabling mechanisms to ambidexterity. We proceed to discuss these results in the light of earlier work. The final section concludes by summarizing the contributions of this paper.

2 LITERATURE REVIEW

In line with the research objectives, we conduct an extant review of research on ADD. This is followed by a review of the contingencies induced by software product development engagements to the ADD setting. We conclude with an analysis of the literature on the theoretical framework of ambidexterity, which guides our study.

2.1 Agile Distributed Development

The focus of the literature on ADD has been to explore the dynamics of agile principles and practices within a distributed setup. Table 1 classifies this literature along three major themes:

Theme	Research Focus
#1	Challenges that arise due to the distributed nature of ADD teams; the role of agile principles in overcoming them (e.g., Dullemond et al. 2009)
#2	Adaptations to agile practices, specifically Scrum & eXtreme Programming (XP), such as *Scrum of Scrums, Remote Pairing* to suit the ADD setting (e.g., Šmite et al. 2010)
#3	Control (e.g., Persson et al. 2012), communication (e.g., Alzoubi et al. 2016) and coordination (e.g., Hole & Moe 2008) mechanisms appropriate to the ADD setting

Table 1: Major focus across the ADD research themes

A few studies (such as Lee et al. 2006; Ramesh et al. 2006; 2012) provide a preliminary picture of the inherent conflict between flexibility and stability. Lee et al. (2006) suggest that agile methods, which demand flexibility, must be adjusted to embrace increased discipline and rigor in distributed software development. Along similar lines, Ramesh et al. (2006; 2012) suggest 'balanced practices' that enable the software team to manage flexibility-stability trade offs. These studies have primarily considered software services engagements, which directly involve the real customer at the development stage. However, we have not come across studies that emphasize the nature of engagement, i.e. services or product development, while examing the accommodations that have been devised by ADD teams.

2.2 Software Product Development Engagement

Generically, a software product refers to off-the-shelf software packages for the mass markets. Sawyer et al. (1999) identify the problems of disconnect with the real customer for packaged software, and suggest modifications to engineer product requirements in this domain. They maintain that the development organization, which is the primary stakeholder, must be willing to engage with the market over the long term, and source new requirements from it on an incremental basis. The software product development possesses an iterative character, where the subsequent versions of the product comprise of enhancements to the existing version and are frequently delivered to the market through well-defined release cycles. The success of such a market-driven project is measured in terms of "sales, revenue, market growth and the ability to create flexible product architecture that can support future releases" (Fogelström et al. 2010).

Implications for ADD Practice

Table 2 highlights the implications of agile on market-driven product engagements, in the developmental context of ADD.

Parameter	Agile Philosophy	Product Development Perspective	Implication for ADD setup
	(adapted from Fogelström et al. 2010)		
Notion of Customer	Bespoke (custom software) perspective arising from a client-vendor partnership	Wide market base; consisting of multiple customers	Providing adequate customer perspective (through proxy groups) across sites
Requirements Elicitation	Evolved through constant customer-developer collaboration	Development organization itself arrives at the requirements after a careful analysis of market inputs and their priorities	Each site is responsible to source requirements for their assigned function
Understanding of Value	Straightforward, through regular customer feedback	Complex, through multiple tools such as Gap and Customer value analysis	Each site must assess the actual delivered value for a release

Table 2: Key parameters for software engagement

At an aggregate level, there is a lack of alignment between philosophical tenets of agile and the needs of product development. Rather than be perceived as risks to execute projects, the product development perspectives represent essential constraints around which the practices of ADD have to be crafted. For instance, a proxy customer group plays the dual role of gauging the right expectations from the end users and accordingly guiding the software team during the course of product development. The success of the effort is dictated by the abilities of this group and the directions offered to the executing team. Such a constraint is accentuated further in a distributed setup, where the proxy customer group would either be splintered across sites or its representatives stationed at specific sites.

A holistic view of the hindrances faced in an agile distributed product development setting is crucial to draw implications for project execution in the domain. From the review of literature on ADD, we infer that the adaptations to agile have been discussed in the greater context of distributed development. We believe that narrowing our focus to the product development space would be of significant interest to the community.

2.3 Theoretical Basis: Ambidexterity

Ambidexterity, in the literal sense, refers to the state when an individual can use both the hands to equal effect. While ambidextrous individuals are able to carry off special tasks, they struggle with trade-off decisions. This notion has been extended to organizational studies since several decades. Firms generally strive for two different things at the same time in their bid for survival as well as continued growth: (1) *exploring* new markets as well as *exploiting* existing assets and capabilities, or (2) *aligning* with current business demands as well as *adapting* to future environmental changes. Ambidexterity relates to the specific ways adopted by organizations to handle the duality, such as the creation of separate business units (Tushman & O'Reilly, 1996) or the provision of an appropriate context for a particular unit (Gibson & Birkinshaw, 2004).

The literature has mainly focused on two types of ambidexterity trade-offs: (1) exploration-exploitation at an organizational level, and (2) alignment-adaptability, also referred to as stability-flexibility (e.g., Vinekar et al. 2006), at a business unit/team level. As a response to mitigate these trade-offs, the literature has presented two major forms of ambidexterity: (1) Structural (Tushman & O'Reilly 1996), and (2) Contextual (Gibson & Birkinshaw 2004). Several of the exploration-exploitation trade-offs have been managed by following the structural approaches, i.e. creation of "dual structures" (e.g., Benner & Tushman 2003; Tushman & O'Reilly 1996) where each sub-unit handles one part of the duality. This approach does not have implications for ADD, which demands the tenets of agile and distributedness in a single project.

Most of the alignment-adaptability trade-offs have been handled by following the contextual approaches, viz. favourable context of a business unit enabling the adequate processes (e.g., Cao et al. 2013; Ramesh et al. 2012). This approach demands a project context, which is characterized by performance management as well as social elements, for individuals to carry out their work. Performance elements are defined by the behavior-framing attributes of *discipline* and *stretch*, whereas social elements are described via the attributes of *support* and *trust* (Gibson & Birkinshaw, 2004). On the one hand, 'discipline' induces the team members to voluntarily strive to meet all the expectations generated by their explicit or implicit commitments, while 'stretch' induces them to voluntarily strive for more, rather than less, ambitious objectives. On the other hand, 'support' induces the members to lend assistance and countenance to others, while 'trust' induces them to rely on the commitments of each other. The interaction between these variables results in ambidextrous capabilities. This approach has direct implications for the ADD setting, where the project context for software teams should allow them to embrace agile and distributedness together.

The main discourse in the ambidexterity literature continues to be regarding how such a capability is developed. Despite an over-arching emphasis on the structural and contextual strategic solutions, "few studies have explained what people actually do to accomplish ambidexterity" (Huang et al. 2014). Recent studies have placed an added emphasis on uncovering the enabling mechanisms to ambidexterity, rather than restricting the focus to strategic choices (structural vs. contextual). Huang et al. (2014) present evidence for *site-shifting* to explain the creation of ambidexterity. They describe how a particular site gets shifted over time, through the development of a relationship between IT-related practices and practitioners. In a similar vein, Gregory et al. (2015) conduct a micro-foundational level study to explain the paradoxes faced by the managers in IT transformation programs, and how they deal with them. In line with the call by Nosella et al. (2012), a clear gap exists to analyze ambidexterity at a micro level, through the lens of organizational practices and routines. This research takes a 'practice-centered' approach to bring out a nuanced understanding of the mechanisms followed by the software teams to mitigate the conflicting forces.

3 RESEARCH DESIGN

We adopt a multiple case study research approach (Yin 2003) with embedded design (multiple units of analysis per case) to understand the complex phenomenon of ADD. This allows us to examine the research questions in their natural setting (Benbasat et al. 1987). We follow a positivist approach informed by prior concepts and theory (i.e., contextual ambidexterity) identified from the literature.

3.1 Data Sources

The study involves two on-going software product development projects (see Table 3) that have progressed well since their inception, and are being executed in an ADD format. These efforts exhibit significant agile process and project maturity. These product engagements involve between 4 and 8 agile teams, working as clusters at independent sites. Both the projects have one common geographical site (India), which results in similar base criteria, and a significant cluster of teams spread across similar time zone differences (e.g. IST and CET). This helps us control for the spatial and temporal aspects of distributedness. We choose two revelatory cases which are diverse for the sake of generalizability to the context. A brief description of the case projects follows.

Proj ChipSys

This ADD effort is focused on developing software for a System-on-Chip (SoC) solution, which is aimed at enhancing existing capabilities for a wireless and mobile platform. Typical users are the leading mobile manufacturers, who in turn develop the end product (i.e. mobile phones) for the consumer market. The solution is a technological improvement with additional features over the predecessor version. It involves more than half-a-dozen cross-functional teams (XFTs) that are working on independent modules. Each XFT is self-contained, and involves multiple teams to address a set of capabilities. For instance, the multimedia module requires display/graphics, camera and video capabilities. Dedicated software teams for each of these capabilities work from three independent locations across Asia and Europe. This allows software teams to work on source code in parallel, and later to merge the code once they are done. For the sake of our study, we shall classify this work breakdown as *autonomous*.

The customer view is derived by collaborating between technical leads and quality assurance personnel that are geographically dispersed and a collocated verification team; they seek and respond to inputs from marketing representatives that are spread across the globe. The software teams at a given site take complete ownership of the components that they develop. The teams' abstract required information with the other XFTs. To conduct the research investigation, we have interacted with the teams based in India that handle display/graphics component. In addition, we have conducted a round of discussion with the team members that are present at the other two locations.

Proj HealthSys

This ADD effort is focused on developing a "health cloud", which is an end-to-end solution for healthcare imaging applications. Typical users are medical practitioners and patients, who wish to share medical information. The solution helps maintain patient history in the form of reports and images, which are especially useful when the patient decides to switch from one healthcare provider to another. Because the information will be stored on the cloud, the patient can easily direct the new doctor towards his/her medical history. Project work involves building the platform that serves as the infrastructure, as well as developing various applications that run on top of it. Software teams from a particular site work on an entire layer of the architecture.

In contrast with Proj ChipSys, there are significant dependencies, because teams that are "upstream" depend on the functionalities being worked on by teams that are "downstream" in the architecture. For instance, the application layer depends on the services layer. This results in a tremendous push for additional planning. For the sake of our study, we shall term this work breakdown as *inter-linked*.

The customer view is crafted by a dedicated portfolio management group (PMG), which is run by the portfolio lead and a set of managers. To conduct the research investigation, we have interacted with the teams from India that operate on the services layer. In addition, we have conducted a round of discussion with distributed team members in Europe.

Characteristic	**Project Details**	
	Proj ChipSys	Proj HealthSys
Domain	Semiconductor	Healthcare
Solution	System-on-Chip (SoC)	Industrial Cloud Based Platform (PaaS)
Motivation to go agile	Internal to a particular XFT	Enterprise-wide agile mandate
Objective for choosing agile	Time-to-market considerations	Short release cycles
Iterations	Release: Depends on the functionality; Sprint: 2 weeks	Release: 3 months; Sprint: 2 weeks
Team Size	Four Scrum teams of 6-8 people working from three sites	Eight Scrum teams of 6-8 people working from three sites
Geographical team split *(Parentheses denote number of teams at a given site)*	India (2) – Germany (1) – China (1)	India (3) – Hungary (3) – France (2)

Table 3: Summary of Case Project Characteristics

3.2 Data Collection & Analysis Technique

We have collected data in a staggered manner across the projects. At any given point in time, the focus has been restricted to a single project. Semi-structured interviews form the primary source of data, each lasting for 45-60 minutes. The interviewer has tried to understand challenges that are faced in the given setting, the nature of interactions with distributed teams and managerial practices that are followed at the site. Respondents have diverse roles: manager, multiple development leads, scrum master, product owner, and developer. In Proj ChipSys, the execution lead and technical lead have themselves played the role of scrum master and product owner respectively. In Proj HealthSys, we have come across additional roles, such as value stream engineer, release train engineer, and portfolio members. An interview guideline has been developed for each of these profiles. Follow-up interviews are conducted to gather additional information after analyzing earlier evidence. Besides this, other sources of data include participant correspondence, site observations and project artifacts. These multiple sources have helped triangulate our data and converge on a single explanation. The validity and reliability of the data are assessed following the guidelines in Yin (2003).

We employ the *Atlas.ti* data analysis software to develop codes from the collected data. We follow coding techniques suggested by Corbin & Strauss (2008), for theory development. Constant comparative method is deployed to ensure repeated comparison of newly collected data with the previous one. Each analysis is documented in the form of a report, which serves as inputs for cross-case analysis.

4 FINDINGS

One aspect is common to the ADD setup across our projects: multiple teams operating at independent sites that are separated by geographical boundaries.

4.1 Conflict: Flexibility-Stability

The case settings reveal that the type of distributed configuration (autonomous versus inter-linked teams) has a direct bearing on the demand for *stability*. Because distributed teams tend to seek increased clarity about project objectives upfront, the ways in which they are structured has an impact on their work execution goals.

We also witness that the mode of engagement, i.e. product development, has a direct effect on the need for *flexibility*. Whereas agile emphasizes constant customer involvement, our case settings are guided by proxy groups. Subsequently, the onus lies on internal software teams to negotiate and reach a consensus on issues, as well as to seek regular feedback from the proxy customers on the work backlog.

Proj ChipSys: Autonomous split

Given an autonomous configuration, *stability* manifests in software teams trying to establish a strong work identity as a specialized unit at an independent site. It announces 'what they are known for in the overall program'. In the words of a Software Project Manager (SPM):

> "*We are emphasizing to the organization that we want to have a Centre of Excellence (CoE); the CoE will be the key focus for a business unit and a particular location."*

A conscious attempt is made to reap some of the benefits from the geographical spread of the teams. ChipSys ensures that the teams are located (1) closer to market and/or (2) receive benefit from core competencies of individuals from a particular region. As remarked by the Execution Lead:

> "*Our ideal preference is co-location/single location but we do not want to lose the advantage of getting a diversified view when the teams are spread. We try to achieve a balance."*

The team division is such that one entire component - display/graphics - is comprehensively handled in India, whereas other components, such as camera and video, are handled at distinct locations in Europe and Asia respectively.

Flexibility manifests when teams must manage volatile dependencies, primarily from an ever-changing market. A lack of real customer involvement results in requirements being indirectly conveyed and constantly reviewed by the proxy customer; multiple change requests on a given module cause frequent re-alignments of software teams. To ensure smooth progress, component teams have to come together on a sporadic rather than a planned basis. As explained by the Execution Lead:

> *"Across components, there may be a dependency, like both **camera** and **video** have to send buffers of the video content to **display**. So, we set up a specific interface, in the form of an android framework. And there are pre-defined rules: What is the other team's exit criterion? What is our entry criterion? We stitch our pieces, and that's where the validation team and our team test [...]"*

Such a level of reliance on another team's activity means that any slippages might affect the work of the teams involved. If dependencies are not handled properly, then it will delay work movement. Moreover, volatile requirements impact the alignment of work of one XFT with other XFTs that handle primary modules. As an illustration, **memory** is the central building block that directly impacts how multimedia aspects of the SoC function; any change to the specifications of the memory module by a particular XFT affects the work of several XFTs, including multimedia. In such a scenario, it becomes critical to follow specific agile practices in spirit. As explained by the SPM:

> *"The point here is: the discussion on deliverables, during sprint planning of other XFTs, may actually evolve. They have a daily alignment already as part of a particular XFT. They are on continuous integration. How will we align cross-XFTs dependencies?"*

Due to such alignment challenges, our interactions reveal instances when the software teams are not able to keep up with their commitments. There are slippages in meeting the target date, while making room for accommodating the outputs of other software teams. The outputs are often received when the latter team is involved with other work. Despite their commitment to the sprint and dedicated focus as a specialized unit, the team cannot push the activity on the received work beyond a particular timeline. They end up creating bandwidth to receive the work and take the task forward, by dropping tasks that have been planned.

Proj HealthSys: Inter-linked split

In this distributed configuration, *stability* concerns multiple software teams working from an independent site striving towards cohesively managing a layer of the architecture. We categorize the teams as "producers" who are responsible to provide the infrastructure, which "consumers" leverage to develop applications.

Collectively, the India teams (producers) work as a single collaborative unit. As the Functional Lead explains:

> *"We are open to share expertise across the teams that sit here and ensure that all teams are brought to a level, where queries can be handled independently by anyone."*

The Scrum Master highlights another evidence in support of cohesion:

> *"We [software teams sitting in India] have separate backlogs to work on. However, we work as a cluster; once there was an instance when two people [developers] were pulled from one team and asked to work on another team."*

Subsequently, the software teams ensure that knowledge is distributed across teams at a particular site. In the words of the Technical Lead:

> *"There are 3 teams here that are responsible for handling storing and retrieving operations. They keep interchanging the responsibilities across iterations so that there is no knowledge silos created."*

In this mode of ADD in product development, *flexibility* again concerns software teams managing volatile requirements due to the presence of the proxy customer (in the form of PMG). The Product Owner of the producer team considers consumer team priorities and interacts with the PMG to refine the backlog. In his words:

> *"Definitely, it [backlog prioritization] is from the workflow perspective – consumer team requirements are top priority. The portfolio group also comes up with a list of prioritized features. Those two together will help us decide what features we have to work on. Based on the bandwidth, we will identify the next set of features."*

Although the expectations are well defined through consultations with the PMG at the iteration planning stage, change requests that arise while an iteration is underway are inevitable. In the words of the Value Stream Engineer:

> *"Change will always be there. But, we do control it. Change requests from PMG go through a tough discussion and conscious effort is made to minimise the impact of changes on teams."*

During the instances of such change requests, the producer team faces the burden of re-aligning their work to absorb changes. The Scrum Master remarks:

> *"Change requests will keep coming in. If it is something urgent, we take a call depending on our bandwidth. Product Owners will be there – they will clarify what needs to be committed."*

Consequently, the setting places an added thrust on the consumer teams to ensure that they regularly recognize the work done by the producer teams. This enables the software development work to proceed in parallel, and also to embrace the contingencies of the change requests.

4.2 Achieving a balance between Flexibility-Stability

In a bid for software teams to achieve a balance between the conflicting forces, ambidexterity is seen as a natural response (Ramesh et al. 2012). Based on the guiding theoretical frame of contextual ambidexterity (Gibson & Birkinshaw 2004) and its applicability to the ADD setting (cf. Ramesh et al. 2012), we elaborate on the project context elements from the case settings. The discussion is interspersed with examples of managerial practices that enable these elements.

As highlighted in our literature review, the project context consists of performance ("hard") as well as social ("soft") elements. In our case settings, we observe that the performance elements (characterized by *discipline* and *stretch*) manifest in the form of the *extent of role specification* within software teams. We also witness that the social elements (characterized by *support* and *trust*) surface through the *nature of boundary spanning* across sites.

Overall, we observe that the case settings achieve a balance between flexibility-stability through an interaction of "hard" and "soft" elements.

Proj ChipSys

The case setting specifies sharp roles for team individuals. This allows each individual to take specific responsibilities within the software team. For instance, software project manager is responsible to track the status of project and coordinate across XFTs, whereas execution lead handles the responsibility to look after the overall work progression by the particular XFT.

The setting places an emphasis on the entire software team members to span the cross-site boundary. They also have the autonomy to use their own judgment to overcome obstacles, including coordination conflicts. As the Product Owner explains:

> *"In most cases, individuals knows who to talk to, and do whatever is required to solve an issue; it doesn't require micro-management."*

Overall, the setting presents ample evidence for the existence of sharply defined roles and responsibilities *yet* provides freedom and autonomy at multiple levels for individuals to take boundary-spanning decisions.

There are several managerial practices that enable such a context. Consider the practice of *planning* where teams ensure that a feasible amount of work that can be completed is taken up for a particular sprint: although the execution lead controls the flow of stories, team members conveniently update and/or add new stories to the board depending on the contingencies that have already been discussed with the teams at different sites, in a bid to accommodate change requests from the proxy customer.

Another practice that supports the specified context concerns *joint de-bugging sessions*: although the execution and component leads lay out the road map for the integration activity, software teams - specifically developers - jointly work on handling integration issues. Additionally, developers have the freedom to organize one-on-one sessions with the counterparts from the other software team if a critical feature needs to be discussed. In the words of a developer:

> *"We have some milestones for a feature. If we have inter-dependency, then we can schedule a periodic meeting with the other team [whoever is working] to finalize the feature and status update."*

Proj HealthSys

The case setting relies on overlap in role specifications for key positions such as the manager and the scrum master; multiple individuals share the responsibility within the team. For instance, the overlap is evident in the designations such as the value stream engineer and the engineering manager. Both are responsible to ensure that the project progresses as per plan. Additionally, the presence of release train engineer, who is responsible for planning work in a given iteration and ensuring that dependencies are ably created, communicated and met, liaisons with the scrum master to address impediments faced by the teams.

The setting provides an evidence for a designated boundary spanner individual. The scrum masters at an independent site discuss project matters between themselves, but the responsibility to coordinate with the other site lies with the release train engineer. As the Engineering Manager suggests:

> "*There is a local Scrum of Scrums, where the scrum masters from a given site participate. They may not be plugged in so much into the global team. But, the release train engineer is very much in alignment with the global team."*

Overall, the setting demonstrates significant overlap in roles and responsibilities across several team members *but* the presence of a specific boundary spanner individual across sites.

There are several managerial practices that enable such a context. For instance, consider the practice of *Scrum of Scrums (SoS) and Release Train Engineer (RTE) Sync-up*: SoS is held locally at a given site, where all the scrum masters come together and discuss progress of the individual team. Subsequently, RTE synthesizes the discussion and takes it forward. In essence, most of the team members share responsibility over SoS, while the RTE gets in touch with the global counterpart across site. RTE's work in close collaboration throughout the sprint. This view is further substantiated through our discussion with the local RTE:

> *"We do have a regular sync-up, like we used to have twice a week earlier, but it's not like we only stick to those timings. For e.g., as and when I have something, we just get into a call and close it."*

Review practice where teams demonstrate their work and get feedback from the proxy customer group also enables the specified project context: several members across the teams at a given site take the onus to prepare the demo presentation, whereas one designated individual presents to a global audience (PMG and business leaders) during the meeting. Subsequently, the feedback is received contingent upon the state of the project and refined market expectations.

5 DISCUSSION

Ambidexterity theory has been recognised as a valuable approach to understand the dynamics of conflict faced by organizations, business units and/or teams. In the field of software development, limited research (such as Lee et al 2006; Ramesh et al 2006; 2012) has attempted to provide clarity on the nature of conflict faced, as well as the mitigation techniques adopted by the teams. We discuss our research in relation to these works.

Lee et al. (2006) focus on coping strategies realized by the software teams to mitigate conflicting demands faced in globally distributed setup. They characterize these strategies along two dimensions: (1) whether the strategy concerned initiation or execution phase of the project lifecycle, and (2) whether the strategy focused on task-related, people-related or technology-related aspects. Ramesh et al. (2006; 2012) suggest practices in the form of tangible and intangible processes, grouped such that they represent specific mitigation strategies to the conflicting demands faced in ADD setting. They remain focused on mapping such strategies to the well-known antecedents (discipline, stretch, support and trust) to ambidexterity.

Our work is based on these predecessors. However, rather than directly focusing on the strategies adopted by the software teams, we have chosen, as our starting point of analysis, the notion of project context. We elucidate the project context for the setting; rather than continue to restrict attention to the general variables of discipline, stretch, support and trust. In this regard, our research brings forward the concepts of (1) extent of role specification and (2) nature of boundary spanning. The case settings reveal that the interaction between these two concepts characterizes a particular ADD project context. We provide exemplars for how these concepts are enabled by the managerial practices, which allows the software team to achieve a balance between conflicting forces present in the setting.

6 CONCLUSION

The contribution of this paper lies in providing insights beyond the earlier conceptualization of flexibility-stability conflict faced in the ADD setting. Our research reveals that the domain of engagement, i.e. product development in this case, drives the demand for *flexibility*, whereas specific distributed team configuration, autonomous vis-à-vis inter-linked split, drives the need for *stability*. Furthermore, we elucidate the concept of contextual ambidexterity for the ADD setting by explaining what software teams actually do to achieve a balance between the conflicting demands. At a broad level, empirical data from the two case projects suggests that the project context, characterized by the concepts of role specification and boundary spanning, which is enabled through some of the managerial practices followed by the software teams lead to development of ambidextrous capability. The findings should provide an exemplar to assist future ADD implementations in practice.

7 References

Agile Alliance. 2001. “Agile manifesto & principles.” http://agilemanifesto.org/ Retrieved 1 July, 2016.

Alzoubi, Y.I., Gill, A.Q., & Al-Ani, A. 2016. “Empirical studies of geographically distributed agile development communication challenges: A systematic review,” *Information & Management* (53:1), pp. 22-37.

Benbasat, I., Goldstein, D.K., & Mead, M. 1987. “The case research strategy in studies of information systems,” *MIS quarterly* (11:3), pp. 369-386.

Benner, M.J., & Tushman, M.L. 2003. “Exploitation, exploration, and process management: The productivity dilemma revisited,” *Academy of management review* (28:2), pp. 238-256.

Cao, L., Mohan, K., Ramesh, B., & Sarkar, S. 2013. “Evolution of governance: achieving ambidexterity in IT outsourcing,” *Journal of Management Information Systems* (30:3), pp. 115-140.

Corbin, J., & Strauss, A. 2008. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Thousand Oaks: California, CA, USA.

Dullemond, K., van Gameren, B., & Van Solingen, R. 2009. "How technological support can enable advantages of agile software development in a GSE setting," in *Fourth IEEE International Conference on Global Software Engineering 2009*, pp. 143-152.

Fogelström, N.D., Gorschek, T., Svahnberg, M., & Olsson, P. 2010. "The impact of agile principles on market driven software product development," *Journal of software maintenance and evolution: Research and practice* (22:1), pp. 53-80.

Gibson, C.B., & Birkinshaw, J. 2004. "The antecedents, consequences, and mediating role of organizational ambidexterity," *Academy of management Journal* (47:2), pp. 209-226.

Gregory, R.W., Keil, M., Muntermann, J., & Mähring, M. 2015. "Paradoxes and the Nature of Ambidexterity in IT Transformation Programs," *Information Systems Research* (26:1), pp. 57-80.

Hole, S., & Moe, N.B. 2008. "A case study of coordination in distributed agile software development," in *European Conference on Software process improvement,* pp. 189-200. Springer Berlin Heidelberg.

Huang, J., Newell, S., Huang, J., & Pan, S.L. 2014. "Site-shifting as the source of ambidexterity: Empirical insights from the field of ticketing," *The Journal of Strategic Information Systems* (23:1), pp. 29-44.

Lee, G., DeLone, W., & Espinosa, J.A. 2006. "Ambidextrous coping strategies in globally distributed software development projects," *Communications of the ACM* (49:10), pp. 35-40.

Sawyer, P., Sommerville, I., & Kotonya, G. 1999. "Improving market-driven RE processes," In *VTT Symposium* Vol. 195, pp. 222-236.

Nosella, A., Cantarello, S., & Filippini, R. 2012. "The intellectual structure of organizational ambidexterity: A bibliographic investigation into the state of the art," *Strategic Organization* (10:4), pp. 450-465.

O'Leary, M.B., & Cummings, J.N. 2007. "The spatial, temporal, and configurational characteristics of geographic dispersion in teams," *MIS quarterly* (31:3), pp. 433-452.

Persson, J.S., Mathiassen, L., & Aaen, I. 2012. "Agile distributed software development: enacting control through media and context," *Information Systems Journal* (22:6), pp. 411-433.

Ramesh, B., Cao, L., Mohan, K., & Xu, P. 2006. "Can distributed software development be agile?" *Communications of the ACM* (49:10), pp. 41-46.

Ramesh, B., Mohan, K., & Cao, L. 2012. "Ambidexterity in agile distributed development: an empirical investigation," *Information Systems Research* (23:2), pp. 323-339.

Shrivastava, S.V., & Date, H. 2010. "Distributed Agile Software Development: A Review," *Journal of Computer Science & Engineering* (1:1), pp. 10-17.

Šmite, D., Moe, N.B., & Ågerfalk, P.J. 2010. "Fundamentals of agile distributed software development," in *Agility Across Time and Space*, pp. 3-7. Springer Berlin Heidelberg.

Tushman, M.L., & O'Reilly III, C.A. 1996. "Managing evolutionary and revolutionary change," *California Management Review* (38:4), pp. 8-28.

Vinekar, V., Slinkman, C.W., & Nerur, S. 2006. "Can agile and traditional systems development approaches coexist? An ambidextrous view," *Information systems management* (23:3), pp. 31-42.

Wong, S.S., & Burton, R.M. 2000. "Virtual teams: what are their characteristics, and impact on team performance?" *Computational & mathematical organization theory* (6:4), pp. 339-360.

Yin, R. K. 2003. *Case study research: Design and methods,* (Third edition). Thousand Oaks: Sage.

Copyright