Information Technology and Management (2023) 24:247–266
https://doi.org/10.1007/s10799-022-00372-w

Conflict management in agile distributed development: evidence from product development and services engagements

Ashay Saxena[1] · **Shankar Venkatagiri**[1] · **Rajendra K. Bandi**[1]

Accepted: 21 July 2022 / Published online: 17 August 2022

Abstract
Agile approaches being practised by multiple teams operating remotely are widely adopted for large software development efforts these days. An agile setting is typically characterized by *flexibility*, 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 (ADD) projects results in an inherent conflict that must be reconciled. This article attempts to provide nuanced clarity on the notion of conflict between flexibility and stability and its management across variants of an ADD setup. Through multiple case studies, our findings suggest that the specific mode of agile project engagement and distributed team configuration drives the response to flexibility and stability respectively. Leveraging ambidexterity as a theoretical lens, this study contributes to the literature by providing insights beyond the earlier conceptualization of flexibility-stability conflict for the ADD setting. It considers contextual elements to understand the dynamics of conflicting forces. An empirical contribution of this research is the managerial framework that should assist practice in future implementations.

Keywords Agile distributed development · Ambidexterity · Conflict management · Flexibility · Stability

1 Introduction

Over the last two decades, organizations have come to embrace agile approaches as a 'better way of developing software' [1]. Traditionally, teams developed software employing a sequential approach, such as the Waterfall model. Requirements drawn up front constituted a contract, which was expected to be implemented across time, with periodic milestones bringing the customers and developers together. Feedback loops were lengthy, resulting in rework, wasted resources, and dissatisfied customers. Agile aimed to address the shortcomings of such approaches by emphasizing close customer collaboration over contractual compliance. Studies have established the criticality of customer involvement to the success of an agile project [16, 33].

Agile practices were originally conceptualized and elaborated in the context of a collocated, single team [17]. This was primarily due to the richness that face-to-face communication provided to teams that were expected to collaborate closely on project tasks. Over time, video conferencing tools, capably supported by Internet bandwidth, began to deliver a sense of realism to transactions across geographies, paving the way for businesses to execute their software projects in a distributed mode. Forces of globalization have now made it imperative for organizations to carry out their large-scale engagements from geographically dispersed software development centres [44]. This move is triggered by environmental factors such as faster time-to-market, access to technical resources across the globe, and prevailing wage differences.

However, for a distributed model to work, projects must address a host of configurational issues arising from work that is performed by teams dispersed across sites. Parnas [58] highlights the benefits of modular design for any software. Managerially, teams can function without a constant need to communicate. Changes to modules can be made in a self-contained manner. Besides, modular designs are better

✉ Ashay Saxena
ashay.saxena@iimb.ac.in

Shankar Venkatagiri
shankar@iimb.ac.in

Rajendra K. Bandi
rbandi@iimb.ac.in

[1] Information Systems, Indian Institute of Management Bangalore, Bangalore, India

comprehended by the teams. Baldwin and Clark [5] add that modularization helps manage complexity, parallelize work, and accommodate future uncertainty. In a distributed setting, the software artefact is decoupled into features and modules, which may be independently designed, developed, and tested at sites across the globe, with periodic integration. Alternatively, the functionality can be parceled out into larger components, which are then developed at multiple sites, with a fair degree of dependency. Over time, agile projects have come to employ a mix of these approaches so they can operate in a distributed mode.

Agile Distributed Development (ADD) is a model in which software projects are implemented by geographically distributed teams, following an agile methodology [65]. ADD implementations are more challenged compared to collocated agile efforts, or distributed efforts with a fixed plan; the distributed setting presents an inherent conflict [45, 64, 65]. Software teams prefer working towards a plan that has been decided upfront. However, agility obliges them to handle new and volatile requirements on an ongoing basis. Moreover, distributed projects must weather challenges from spatial, temporal, as well as configurational dispersion across teams [56]. Whereas agile teams require the *flexibility* to continuously update the software artefact and deliver relevant functionality, distributed teams seek *stability* to meet their project objectives in a predictable manner. This conflict is not peculiar to ADD project engagements. Researchers have examined multiple domains with similar conflicting demands between alignment and adaptability [28].

ADD projects operate in different engagement modes, depending on whether the teams are working towards delivering (1) off-the-shelf software for the wide market or (2) customized software for a specific client. ADD engagements are unique in that the flexibility-stability conflict presents itself across the duration of the project; the "infinite malleability" of software artefacts [11], distinguishes it from settings such as manufacturing or semiconductors, where the prospect of modification decreases over time. This makes it worthwhile to explore the nuances of the conflict, and study management approaches that are followed in varied ADD contexts, ranging from software products to services. The context-driven insights that emerge will help software teams better manoeuvre their way through an ADD implementation. This leads us to our first research question:

RQ1: What is the nature of the flexibility-stability conflicts that arise in different forms of ADD engagement?

Research suggests using the lens of organizational ambidexterity to explicate and manage the flexibility-stability conflict within ADD settings (cf. [65]. Ambidexterity refers to the specific ways adopted by an organization and/or a business unit to reconcile the tensions that arise while dealing with conflicting choices. This could involve the creation of separate units or structures to handle the duality [24, 75], alternatively, an appropriate context may be created for a business unit [28]. A closer examination (cf. [69] reveals that the choice of the specific form of ambidexterity to guide the research relies upon: (1) where it needs to be pursued and (2) how it needs to be pursued. The ADD setup demands simultaneous pursuit of the conflicting demands between flexibility and stability in a single project. Hence, a contextual form is appropriate to manage trade-offs, based on a 'project context' in which the teams carry out their work. ADD teams can synthesize their responses to problems that arise by combining formal and informal elements of this context. For instance, a team can have precise role specifications for each individual but provide the freedom to span cross-site boundaries for project related matters. These contextual characteristics have direct implications on the way ADD projects are executed. However, the literature on ADD has seldom focused on such details to derive insights for practice (cf. [38]. Keeping in mind the dichotomy between product development and services projects, our study addresses a second research question:

RQ2: How do software teams manage the conflict between flexibility and stability, across different forms of ADD engagement?

This paper reports the results from field research using an exploratory multiple case study approach [81]. The research includes two cases each of product development and services engagements to examine the flexibility-stability conflict. We also examine variations in team configuration across sites across these projects. Our research contributes to the theory of ambidexterity by extending the notion of 'project context' for a given ADD setting. Prior literature has established that the contingencies experienced by ADD teams due to prevalent contextual elements remain to be examined (cf. [65]. Based on the interplay between these elements, we present a framework that can guide ADD teams in handling the dual demands of flexibility and stability.

The rest of the paper is organized as follows. The next section presents a review of the key literature and theoretical foundations that serve as the basis for our study. This is followed by a section that highlights our research design. Subsequently, we present the findings from the case studies, which address the research questions. The penultimate section discusses the relevance of these findings in relation to earlier works. The article concludes by summarizing the contributions of this research.

2 Literature review

2.1 Agile distributed development

Literature on ADD can be grouped into the following themes: (1) Studies that highlight the interplay between agile principles and the dimensions of distributedness, (2) Studies that present agile practices that are tailored to suit the project context, and (3) Studies that showcase the control, communication and coordination mechanisms followed in these settings.

The first theme highlights the role played by agile principles in tackling challenges that arise due to the distributed nature of software teams. Dullemond et al. [23] suggest that agile principles can alleviate risks due to spatial, temporal, and socio-cultural dispersion; each principle has a concomitant effect on team behaviour. For example, if the teams rely on close collaboration, then this indirectly enhances informal communication and eases initiating contact between teams spread across geographies.

Holmström et al. [35] highlight the positive influence of practices that are enshrined in eXtreme Programming (XP) and Scrum. For instance, pair programming demands developers to spend time together on the codebase, thus reducing the severity of temporal distance. For teams that are spread across geographies, pairing encourages them to increase the time overlap, and leverage the expertise with a specific code module across multiple locations [4, 10, 41]. The findings reveal that ceremonies like Scrum planning enhance the "teamness" among distributed teams. The developer-centric qualities of agile practices are shown to promote communication and collaboration, and reduce the effects of socio-cultural distance. These results pertain to a specific scenario involving dispersion among the members of a single team.

Lee et al. [45] recommend that agile methods must be adjusted to embrace more discipline and rigor in distributed environments. They argue that though agile does not mandate extensive documentation, capturing and sharing tacit knowledge becomes critical in global project contexts. With the geographical separation, formal communication modes such as email and wikis take priority over informal modes such as phone calls and messages.

Ramesh et al. [64, 65] interpret such demands as competing ideologies between the "relaxed" expectations of agile and the realities of a distributed setup, resulting in tensions faced by ADD teams. They suggest a mix of balanced practices that could assist teams in their endeavour to execute ADD efforts in a smooth manner. For instance, developer tools could enable a mix of minimally documented requirements (such as short use cases), these are supplemented with informal communication channels (like a video chat) to coordinate the work.

Building upon these insights, the second theme focuses on the adaptations of agile practices to suit a distributed project context [70]. Individual ADD teams practising Scrum can embrace a formal meeting practice called 'Scrum of Scrums', where representatives of each constituent team come together and discuss their progress [39, 57, 72]. This group will work towards identifying and eliminating inter-team blockers and dependencies.

The agile approach promotes the idea of self-organizing, cross-functional teams. With the added dimension of distributedness, newer mechanisms for control, communication and coordination are in order. The third theme elaborates on these "3Cs". In an ADD context, control refers to scrutinizing the actions taken by the developers to fulfil the expectations of the customer and effectively respond to changing user requirements, [49]. Persson et al. [60] suggest that formal control practices which involve measurement and evaluation of outcomes are predominantly carried out in conjunction with an informal clan-like control in the setting.

The importance of formal as well as informal communication for agile software development is widely recognized [37, 43, 50]. Formal modes include "official meetings during inception, weekly meetings and daily stand-ups, and specification documents from clients" [21]. Other forms of interaction such as casual communication, online chat and short messenger service (SMS) constitute the informal mode. Agile methods rely heavily on informal face-to-face communication, but distributedness mounts challenges ranging from team size and distribution to technology issues [2]. ADD teams usually embrace strategies that involve direct communication between developers and a well-defined customer [42].

Task dependencies and team configuration play a central role in the division of work on software projects. At the level of an individual program, Podgurski and Clarke [61] classify dependencies between statements as syntactic and semantic. Cataldo [15] generalizes these concepts to large-scale software systems; a function call from one module to another links the two syntactically, whereas they are semantically linked if one module can affect the execution behaviour of the other. These linkages have implications on task parallelization and team formation. Scaling and distributing the effort across geographies further complicates this picture. Successful project engagements competently manage interdependencies among team members; coordination refers to the act of managing these dependencies [48]. Agile development relies on informal mechanisms (coordination by mutual adjustment) with the key emphasis on people and creativity. In contrast, distributed software development relies on formal mechanisms (coordination by standardization) to exploit detailed documentation, and address impediments to communication due to the geographical separation [64].

Balancing the two coordinating mechanisms poses a major challenge to using agile in a distributed context [34].

Across all three themes, the research on ADD suggests that adaptations to agile have been discussed in the greater context of distributed development. Empirical research that concerns the handling of inherent conflicts treats 'agile' and 'distributedness' as broad terms, with a diverse perspective on elements of managerial interest. The existing research seems to lack a nuanced discussion around the execution of ADD projects under different kinds of engagement models. Variations across the type of customer involvement present contingencies to the ways in which requirements are elicited and business value is delivered by the software teams from across sites. We believe that narrowing our focus to the specific ADD contexts would be of significant interest to the community.

2.2 Software project engagements

Arora et al. [3] classify software project engagements into "custom developed software and packages or generic software products". Xu and Brinkkemper [79] supply an alternate categorization as tailor-made software and product software. Whereas a single customer forms the focus of tailor-made software efforts, products are created for a broader market. Tailor-made software can be developed either in-house or on a contractual basis with a vendor. Product software can be business-to-business (B2B) or business-to-consumer (B2C). Xu and Brinkkemper clearly differentiate the categories along development lifecycle and methods, requirements and release management, architecture, and delivery and implementation. To illustrate, the architecture for tailor-made software can enjoy more flexibility than for product software, which must take a wide user base as well as multiple versions into account; in addition, it must be "future proof".

Software product engagements involve developing "off-the-shelf software for the mass markets" [62], or enhancing existing packages with new features. Product teams must continually interpret the needs of a vast external market, while soliciting sustained participation from stakeholders within the organization [46]. They must autonomously decide what functionality to include in a version, as well as the timeline to release it to the intended market segments [19, 62, 66]. This presents a need to precisely plan the content of a product release [13, 14, 31]. Organizations must also anticipate future products that influence consumer demands through breakthroughs and innovation. The success of a product development project is measured in terms of "sales, revenue, market growth and the ability to create flexible product architecture that can support future releases" [26].

In this paper, we term tailor-made software efforts as services engagements. Over the decades, the scope of software services has expanded to technology consulting, system integration, custom development, package implementation and application outsourcing maintenance [52]. Stakeholders are expected to observe a key tenet of agile, namely, close customer involvement at every stage of development. While these projects typically service a client team that sits at one location, the software may be developed at multiple centres across the globe. Even in instances when the overarching goals of two efforts are similar, the actual implementation varies from one customer to another, depending on their business processes and information systems [71]. Project success is measured by the fulfilment of customer needs upon the completion of work [26].

With a sharp focus on close customer collaboration and rapid creation of business value, agile methods capably support the perspective of software services projects; such an alignment is lacking in the case of product development projects. The contingencies highlighted above present additional constraints, beyond those faced during agile execution in a distributed setup. A holistic view of these contingencies induced by the unique characteristics of the project engagements is crucial to draw implications for agile execution in distinct contexts. Our research focuses on the inherent conflicts faced by diverse ADD engagements, and their management across both product development and services space.

2.3 Ambidexterity

Ambidexterity refers to the pursuit of conflicting demands that require trade-offs [24]. The most extensively studied trade-off in business literature has been between exploration and exploitation. Typically, the growth prospects of an organization depend on its propensity to 'explore', whereas its immediate survival depends on its capacity to 'exploit'. However, it is difficult to establish an optimal mix of the two [47]. Generally, organizations tend to be biased towards exploitation, given their emphasis on short term success. However, activities that cover both aspects are critical for organizations to prosper [8, 63].

The second important trade-off addressed by ambidexterity studies is between alignment and adaptability. Alignment refers to the actions of an entire business unit pursuing a unified goal. Adaptability refers to its ability to meet changing demands in the task environment. Aside from a few exceptions (see [12, 59], this duality has been examined only at an overall business unit or team level. Business units require a conducive organizational context to carry out their tasks and meet competing demands. Leaders of these units ensure that such a context is created [28].

The alignment-adaptability trade-off has been made central in settings with conflicting objectives. Studies on outsourced software development (e.g. [12, 74] highlight the simultaneous ability of a vendor to adhere to client needs and to address evolving client requirements. *Client needs*

refer to the output expectations from a vendor in terms of cost, scope and quality; these may be specified in a contract. *Evolving client requirements* refer to definitional updates made to the artefact that is being developed. Such settings provide an inherent conflicting demand for alignment and adaptability.

There are two major ambidextrous approaches to manage these trade-offs: (1) Structural [75], and (2) Contextual [28]. Typically, exploration–exploitation trade-offs have been managed by following a structural approach, via the creation of "dual structures"; each sub-unit handles one part of the duality (e.g., [8] and [55, 75, 77]. On the other hand, alignment-adaptability trade-offs have been handled by following a contextual approach, this involves harnessing a favourable context within the business unit itself to enable the adequate processes (e.g., [12, 40, 59, 65, 73, 74].

In ADD settings, Ramesh et al. [65] flag the infeasibility of any structurally ambidextrous response that 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 to handle the problem. This demands a favourable project context, which is characterized by performance management as well as social elements, in order for individuals to carry out their work [68].

Performance elements are defined by the behaviour-framing attributes of *discipline* and *stretch*, whereas social elements are captured by relational attributes of *support* and *trust* [27, 28]. 'Discipline' induces 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. 'Support' induces team 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 building ambidextrous capabilities.

Recent years have witnessed several debates (cf. [7, 36] around the conceptualization of ambidexterity. Benner and Tushman [7] address the critiques and suggest the need to take a closer look at how ambidexterity is exercised in varied settings. In line with their articulation, the main discourse continues to focus on how such a capability is developed. Huang et al. [36] points out that despite an overarching emphasis on the structural and contextual strategies, "few studies have explained what people actually do to accomplish ambidexterity". Through an empirical study, they describe the emergence of practices from being transactional at the onset to being more relational in the subsequent phases. This *site-shifting* over time, through the development of a relationship between IT-related practices and practitioners, explains how ambidexterity is achieved in the setting.

In a similar vein, Gregory et al. [32] conduct a micro-foundational level study to explain the paradoxes faced by managers in IT transformation programs, and how they deal with them. They present managerial responses that involve a mutual blending of IT and business interests to achieve IT transformation objectives. Another empirical study by Zimmermann et al. [82] on corporate innovation initiatives provide evidence for how frontline managers assign less significance to senior managers ambidextrous strategic choices. Instead, the findings suggest that frontline managers prefer to keep the focus on configurational practices to cope with competing forces.

In line with the call by Nosella et al. [54] and recent findings by Zimmermann et al. [82], future research needs to analyze ambidexterity at a micro level, through the lens of organizational practices and routines. Our research takes a 'practice-centered' approach to help develop an understanding of the mechanisms followed by software teams to handle conflicting forces.

3 Research design

To understand the complex phenomenon of agile distributed development (ADD), we adopt a multiple case study approach with embedded design involving multiple sub-units of analysis per case [81]. Our review of the literature reveals that research on conflicts faced by teams in an ADD setting is at a nascent stage. Hence, an exploratory approach appears to be suitable for the study. Multiple case studies increase the methodological rigor by strengthening the precision and validity of the findings [51], allowing cross-case analysis [6] and making the evidence more compelling [80].

Examining multiple projects has helped us arrive at a framework to assist future ADD implementations. While identifying potential conflicts across ADD teams, the framework also suggests concrete actions that lead to better project outcomes in terms of software artefacts and team relationships. The case selection is contingent upon the dual dimensions of agility and distributedness. Following Cataldo's [15] classification based on task modularization for globally distributed software development, we examine 'distributedness' dimension along the lines of autonomous vs inter-linked team split across sites. Besides, we rely upon the long-standing classification (cf. [3, 79] of product development vs services engagements to study 'agile' attribute of customer involvement for our case settings. Table 1 captures these classifications and illustrates the case selection criteria followed for this research.

Consequently, a 2×2 research design (see Fig. 1) will ensure that we involve a significant number of revelatory cases that span diverse ADD contexts.

Table 1 Case selection criteria

Agile attribute	
Type of customer involvement	Direct: An actual customer engages with the development team on project related matters (services engagement) Indirect: A proxy customer, in the form of a group of representatives, provides directions to the development team (product development)
Distributedness attribute	
Configurational differences	Autonomous: Teams working on fairly independent modules across sites Inter-linked: Teams working in a closely coupled mode across sites

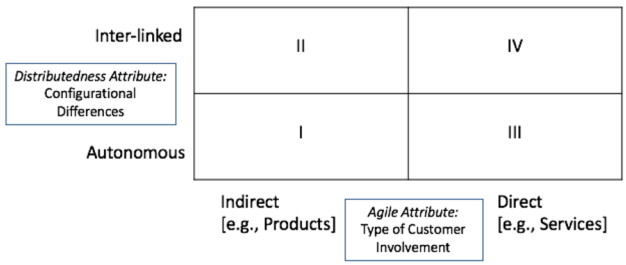

Fig. 1 Research design

Ideally, we must examine one project within each of the four grid cells to serve the purpose of 'theoretical replication' [81]. We could then investigate the variance in our conclusions along a specific agile attribute (e.g., customer involvement) and a distributedness attribute (e.g., configurational differences).

3.1 Project sources

All projects that we have chosen to study have been following agile in a distributed setup since their inception. They all have sufficient maturity, with at least one-quarter of development effort completed at the time of our research investigation. Progress is typically measured on these projects by examining the team's *velocity* and *burndown chart*. All the teams have reasonable agile process experience. These factors enable us to investigate contextual elements, which have stabilized over a period. We maintain homogeneity across projects by selecting engagements that lie within a small range of sizes, between three to ten software subteams. On the other hand, cases from heterogeneous domains throw light on a variety of issues.

To meet case selection criteria, we have reached out to product development as well as services-oriented software organizations. These are multinational corporations (MNCs) spread across geographies, providing a fertile ground to study the effects of distributedness among agile teams. These organizations have facilitated access to multiple projects. Hence, we adopt a "controlled opportunistic" research design, taking advantage of multiple sources of information as and when they present themselves [25].

The four organizations chosen for the research include a semiconductor major, a healthcare IT major, an IT consulting major, and an IT services major. Table 2 summarizes the four cases we have studied across these organizations. Types I through IV refer to the 2×2 classification of Fig. 1.

The **product-focused cases** entail developing software solutions for a wide market. *Proj ChipSys* implements a System-on-Chip (SoC) solution aimed at enhancing some of the existing capabilities for wireless and mobile computing. *Proj HealthSys* focuses on the user and organization management modules for a 'Healthcare Digital' platform, which is an end-to-end solution for imaging applications.

The **services-oriented cases** involve developing applications to suit the needs of a single client. *Proj TrainSys* implements a reservation system for a train aggregator, which sells tickets on behalf of train operating companies. *Proj HotelSys* caters to a website and mobile application requirements for a conglomerate in the hospitality sector.

3.2 Data collection

The sampling strategy ensures that significant projects are selected for insights and variety. We have undertaken data collection across the case organizations in a staggered manner. Purposive sampling has guided our selection of further cases, contingent on the codes that have emerged from a preliminary analysis. This has helped us develop theory on an emergent basis [29]. We have restricted our attention to one case project at a given time; this helps us refine the data collection on an ongoing basis and signal new areas to be explored in subsequent cases.

Semi-structured interviews of 45–60 min duration form the primary source of data collection across the projects. These interviews attempt to unravel the challenges faced by ADD teams in the given project setting, the nature of interactions among the distributed teams, and managerial practices that are followed at each project site. We examine the involvement of diverse roles in the setup, such as project managers, product owner, scrum master and developers. A common guideline has been followed to interview each of these profiles. All interviews are recorded and the transcript, together with the researcher's understanding of

Table 2 Characteristics of software projects selected for the research

	Product engagement		Services engagement	
	Type I (autonomous)	Type II (inter-linked)	Type III (autonomous)	Type IV (inter-linked)
Case(s)	*Proj ChipSys*	*Proj HealthSys*	*Proj HotelSys*	*Proj TrainSys*
Domain	Semiconductor	Healthcare	Hospitality	Transportation
Solution	Mobile System-on-Chip (SoC) solution	Industrial cloud-based platform (PaaS)	Hotel website & Mobile application operations	Train reservation system
Elapsed time on the project as of our research investigation	15 months	7 months	18 months	A few years
Agile process experience	Project driven initiative since inception	Enterprise-wide agile mandate	Project driven initiative since inception	Enterprise-wide agile mandate
Iteration lengths	Sprint: 2 weeks Release: Depends on the functionality	Sprint: 2 weeks Release: Regular (Three months)	Sprint: 2 weeks Release: Component-based, contingent upon teams finishing tasks across layers of architecture	Sprint: 2 weeks Release: Regular (Bi-monthly)
Division of work	4 Scrum teams of 8 people each, working from 3 sites	8 Scrum teams of 6 people each, working from 3 sites	3 Scrum teams of 6 people each, working from 2 sites	10 Scrum teams of 8 people each, working from 2 sites
Team split by geographies	India (2) – Germany (1) – China (1)	India (3) – France (3) – USA (2)	India (2) – USA (1)	India (4) – UK (6)

Table 3 Data Collection across the projects

Organization	Cases	Interviews	Documents	Observations	Project Artefact
Semiconductor major *[10 Site Visits]*	*Proj ChipSys*	13	Process specification; Project manual	Work at India site; Meetings (Sprint Planning)	Virtual kanban board [Chart & XLS]; Sharepoint
Healthcare IT major *[29 Site Visits]*	*Proj HealthSys*	12	Organization beliefs; Project manual	Work at India site	Digital story wall
IT Consulting major *[16 Site Visits]*	*Proj TrainSys*	13	Process specification; Project article	Work at India Site	Digital story Wall; Wiki Page
IT Services major *[10 Site Visits]*	*Proj HotelSys*	11	Project bi-weekly report	Work at India Site; Meetings (Daily scrum of scrums; Sprint planning; Sprint review)	Digital story wall

the setting, are shared with the interviewee to check for any mistakes of commission and omission, and for any incorrect interpretations.

Other sources of data include documents, on-site observations, and a few project artefacts. We have obtained and analyzed documents such as *process specifications* to understand the agile process steps for a given project. Alternatively, a *project manual* helps us ascertain the particulars of the development work as well as team relevant details for the project. We have examined project artefacts such as the *story wall* at project sites. We have also undertaken in situ observations to study how team members collaborate on specific project tasks; these involve attending formal meetings as well as reflecting on team member interactions in the project bay. Tables 3 and 4 provides the exact break-up of the data sources across the case projects.

Barring the semiconductor major, all organizations provided us with a cubicle space in the project bay. Site visits have involved spending thirty minutes in the project bay to observe team interactions. We have passively participated in ceremonies such as sprint planning and review meetings, and have observed cross-site interactions on project related matters. Besides, we have attended a Scrum of Scrums meeting to understand how teams handle cross-site dependencies for a project. We have witnessed at least one daily stand-up meeting on each case to observe the presence or lack of cohesion within a team. During these interactions, we have

Table 4 Details of interviews across each of the case projects

Organization	Cases	Designation of interviewees	Rounds of discussion
Semiconductor major *[10 Site Visits]*	*Proj ChipSys*	Software Project Manager (SPM), Execution Lead (EL) [also Scrum Master], Component Lead #1, #2, #3*, Technical Lead [also Product Owner]*, Developer	SPM & EL (5), SPM (3), Others (1)
Healthcare IT major *[29 Site Visits]*	*Proj HealthSys*	Director Engineering$, Portfolio Lead, Portfolio Manager, Engineering Manager (EM) #1, #2, Agile Coach$, Scrum Master (SM), Technical Lead (TL)	EM#1 (2), EM#2 (2), SM (2), TL (2), Others (1)
IT Consulting major *[16 Site Visits]*	*Proj TrainSys*	Delivery Manager (DM), Iteration Manager (IM), Business Analyst, Quality Assurance Analyst, Technical Architect, Feature Owner [also Developer], Distributed Representative [also Developer]*, Developer	IM (4), DM (3), Others (1)
IT Services major *[10 Site Visits]*	*Proj HotelSys*	Group Project Manager$, Onsite Technical Project Manager*, Project Manager (PM) [also Scrum Master], Quality Assurance Analyst, Test Lead, Technical Lead (TL)	PM (5), TL (2), Others (1)

$Brief round of discussion

*Sits onsite

paid particular attention to the efforts made by team members to alleviate the concerns faced by each other.

These sources have helped us triangulate the data and converge upon a single explanation for a phenomenon such as a conflict; the interviews allow us to form an understanding of the central research themes, which are validated with on-site observations and project artefacts. In addition, the documents provide supplementary information that helps us comprehend the specifics of an ADD effort.

4 Research methodology

This study adopts a neutral and passive perspective [22]. We follow a positivist approach informed by a priori concepts and the theory of contextual ambidexterity. We leverage a computer assisted qualitative data analysis software (CAQDAS) package to develop codes by examining the data gathered from interview transcripts and observations from site visits. The tool furnishes a feature to filter each of the data files, and search for specific keywords.

Coding techniques suggested by Corbin & Strauss [18] and Saldaña [67] have been adopted to form multiple levels of code, from open to selective. The data is organized at three levels:

- A **case** refers to an individual project that is being studied as part of our investigation.
- A **construct** represents a predefined theme based on the central elements of research. The list of constructs includes *Challenges, Conflicts, Practices,* and *Project Context*.
- A **code** corresponds to a symbolic assignment of a summative attribute for a fragment of qualitative data.

We have employed the *Atlas.ti* platform to develop codes from the collected data. The tool facilitates moving across categories and code. We have used it to infer relationships among entities, resulting in the creation of a graphical view of the data. Search operations can be harnessed to browse through relevant quotations that have been tagged with specific codes, enabling the identification of a theme across files, and observing how multiple themes can be interrelated to form a secondary-level code.

4.1 Coding process

We develop an initial set of codes to form preliminary 'categories' around a phenomenon, to highlight important information without making an explicit attempt to distil the categories (open coding). Table 5 provides a sample of preliminary codes.

Subsequently, we establish links based on similarities and differences across the multitude of preliminary codes to form secondary level categories (axial coding). Eventually, we make broad theoretical abstractions through a further refinement of the developed codes (selective coding). The constant comparative method has been deployed to generate theory [30]. This method ensures repeated comparison of newly collected data with the previous one. Table 6 presents an overview of the developed codes from one of the case projects, *Proj HeathSys*. Teams in this product engagement were building a cloud-based PaaS solution, with "producers" in India developing the platform, while "consumers" in Europe & USA focused on front-end applications.

Following the coding procedure, a rigorous within-case analysis has been carried out to develop a sound understanding of each project. We document each within-case analysis in the form of a report, which serves as an input for cross-case examination, to understand variations across

Table 5 Example of organizing data

File	Data	Code
06-HC-H-MGR-22012016\01	*"People are experienced in this [India] team – provides engineering leadership, so are looked upon to take care of the challenges and come up with the right solutions."*	Stability: Specialist team members
06-HC-H-MGR-22012016\13	*"Here we have senior people – they think about the customer, business value—so most of the time they will understand [a change request] and take it further. If a change is not acceptable, something is not logical, then we are all involved to discuss and decide whether we take it up or not."*	Project context: Discipline

the projects. The findings from these investigations provide insights on the research themes.

5 Findings

This section synthesizes evidence from multiple cases to elucidate the conflict between flexibility and stability and its management for different ADD settings. We begin by discussing field observations concerning variations in response to (1) a demand for flexibility and (2) the need for stability.

5.1 Demand for flexibility

We observe distinct differences in handling the demand for *flexibility* in product development and services projects. Product-focused cases are heavily reliant on ever-evolving market dynamics, and their influence on the way teams handle change requests. Notably, the teams agree to craft and implement feasible refinements from market preferences that are managed and enunciated by a proxy customer. In contrast, services-oriented cases rely on an onsite client offering clear guidance based on the impact on cross-site coordination. These teams exhibit a more welcoming and committed handling of changing client expectations. The direct involvement of a real customer in the software development process results in a greater push for embracing change requests on an urgent basis.

5.1.1 Type A: Product Development

Proj ChipSys has an autonomous configuration, with teams working on fairly independent modules across multiple sites. The project has strong involvement by marketing representatives; these proxy customers regularly update the teams on market expectations, which invariably involve change requests to the features under development. The Software Project Manager (SPM) explains how such requests are handled:

> *"A request originates from the customer through marketing [representatives]. It will be fed back to the cross-functional teams (XFT) to carry out a feasibility study. Once the feasibility of the request has been approved, it comes back as an official backlog item to development."*

In acknowledgement of the inputs, each software team may choose to realign their execution.

Proj HealthSys has an interlinked configuration, with teams working in a closely coupled mode across sites. A centralized product management group drives requirement changes, assisting teams across multiple sites by conducting a feasibility analysis of any modifications to their work. The Portfolio Manager who has oversight across geographies explains:

> *"Once the release level planning is done, I resume working on the backlog roadmap, and assess the feasibility margins for execution. Usually, I engage product owners, architects, and engineering managers to itemise what needs to be done, and to check whether everyone agrees with the items on the list or not.*"

Upon receiving the updated requirements, the software teams attempt to realign their priorities and coordinate across sites, so they can deliver business value as a combined unit.

5.1.2 Type B: Services

Proj HotelSys has an autonomous configuration and must often handle ad-hoc requests from the client. As the Scrum Master explains:

> *"[…] Because of the services industry, we must bend to requests. Even though I don't have the capacity due to existing priorities, I may have to stay back and deliver it. Several such instances have occurred. Especially, this sprint has been very tight – some people worked on weekends too. It's not something we had planned, but because of the late arriving work, we did it."*

Table 6 Coding protocol

Construct	Initial coding strategy	Examples from case study [Proj HealthSys]	Developed Code
Conflict	Flexibility: Focus on the data chunk that represents the ability of software teams to provide continuous delivery of a working artefact Stability: Focus on the data chunk that represents the need of software teams to predictably meet project objectives in the setting	Scrum master: *"It becomes a challenge when we as producers [platform team] keep changing things from our side [owing to pivots in project expectations], and the consumer team [application team] is not yet ready to adopt it. We want them to adopt it as early as possible. Also, if we change something or miss anything during the impact analysis, then we place added emphasis [i.e. we may have to add or remove certain stories] to ensure that the workflow is not broken."* Project manager: *"Collectively, teams [at a given site] are responsible for delivering the code. We want self-sustaining teams. We are going in that direction. Also, teams themselves come up with their areas of improvement. Managers only get involved to understand and explicate more, rather than prescribing things."*	Flexibility: Managing volatile dependencies due to an ever changing market Stability: Cohesively managing work at a given site
Contextual ambidexterity	Assess the extent to which teams simultaneously demonstrate flexibility* and embrace the need for stability* in the setting Note: 'High' on both these dimensions (*) signifies ambidexterity [53]	Project manager: On an average [over the last 5–6 sprints]: How often has the software team adhered to accomplish project objectives determined for the site? Ambidextrous capability Never Rarely Sometimes Often Always ✓ How often has the software team been able to accommodate changes to the project plan? Never Rarely Sometimes Often Always ✓	
Project context	Multiple levels: [Performance elements] Discipline Stretch [Social elements] Support Trust	Engineering manager: *"Release Train Engineer and Scrum Master are supposed to look after multiple things: (1) Day-to-day perspective of how things are progressing, (2) Managing dependencies, and (3) Flagging risks at the right time"* Technical lead: *"Once technical challenges on a new API requirement is identified, there will be a small group consisting of architects, tech leads, and product owners that will meet formally over WebEx. The group will identify the specifics of the requirement and suggest changes to the existing plan."* Scrum master: *"Basically, if it's something about a dependency or schedule, it will come to the Release Train Engineer. If it is about a clarification/change in requirement or any impact analysis that we have done for our change, it will again be through him."* Engineering manager: *"We try to build a dependency map and see who is dependent on whom, and what you need from them. For example, if I need something from a different team, then during the release planning, we will make a dependency map – put a sticker and explicitly mention the date by which I need the deliverable. So, it is very clear for the other team."*	Discipline: Clarity in role specification Stretch: Shared discretionary work effort Support: Designated boundary spanner individual Trust: Visibility of work progression

Such unplanned requests result in an intermittent reliance across the sites to complete work. The onsite team lends a supporting hand, and shares responsibility with the various offshore teams to ensure that the updated requests are developed on time.

Proj TrainSys with an inter-linked configuration is obliged to comply with the client's expectations on an ongoing basis. The client in this case is a Train Operating Company (TOC). The Delivery Manager, who keeps track of the progress on work deliverables, remarks:

> *"Anything that originates from the client's end leaves us with little choice. We just have to do it. The India team owns the vertical. If the client comes back saying, they need this feature and they are willing to pay for it, and it's revenue for their business, then we will do it."*

The interlinked mode of engagement warrants continuous task coordination across sites to fulfil project expectations. With each instance of a change request, *Proj TrainSys* relies on dedicated representation from across sites to ensure that client demands are adequately met.

Overall, we see that product-focused cases rely upon software teams implementing *feasible updates from evolving market preferences*, whereas the services-oriented cases involve more *committed handling of client expectations* by the executing teams.

In light of our findings, we propose the following:

P1: Given an ADD setting, the mode of agile project engagement—whether there is indirect or direct customer involvement—has a direct bearing on the response to demand for flexibility

5.2 Need for stability

Our case settings reveal that an autonomous versus interlinked work breakdown across sites results in distinct handling of the need for *stability*. This time, we will contrast projects in the two ADD team configurations.

5.2.1 Configuration I: Autonomous

Operating in a product development mode, *Proj ChipSys* wishes to ensure that a component is comprehensively managed out of a given site. In the words of the 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."*

In the services-oriented setting, *Proj HotelSys* involves independent teams that work on a specific module or function from a given site. As the business analyst remarks:

> *"The reason we have identified individual teams and given them specific responsibilities is to make things simpler. To achieve the greater cause, the first responsibility of the team is to achieve their own targets. Once that is accomplished, then probably they can look outward and ask - now that we have achieved our goals, is there something we can do to help others?"*

The software teams rely on their independent responsibilities to manage overall commitments for a given site. This also enables specific sites to control overheads, such as coordination costs, which could adversely affect the progress of the project.

5.2.2 Configuration II: Inter-linked

In the product development setting, *Proj HealthSys* has one site acting as a "producer" responsible to provide infrastructure, which "consumers" leverage to develop their applications. An engineering manager explains the mode of working at the India site:

> *"Since the platform [teams in India] needs to cater to multiple applications [teams in France and USA] at the same time with a similar set of capabilities, the teams here prefer to work jointly to ensure that site-level commitments are handled adequately."*

Any slippage on delivering a dependent piece of work on time from a given site results in overall project delays. Therefore, the teams at a given site prefer to support each other in their endeavour to complete and deliver work to their counterparts as per the committed timeline.

In the services-oriented setting, *Proj TrainSys* has the front-end and back-end work addressed by different sites across geographies, with occasional overlaps. The teams again prefer to ensure that they work collaboratively at a given site to manage their commitments. As the delivery manager states:

> *"Every development site [involving multiple teams] will manage all stages, starting from the conceptualization of a feature until it goes live. The entire lifecycle is managed collaboratively by the respective unit."*

Overall, we witness that projects with an autonomous work breakdown across development sites have teams working in a self-contained mode at a given site to comprehensively handle a component or a feature, whereas projects involving inter-linked work breakdown across development sites involve teams working in a collaborative manner at a given site to ensure that dependencies are adequately met.

In the light of our findings, we propose the following:

Table 7 Conflict between flexibility and stability

	Product development	Services
Autonomous	Software teams work in a self-contained mode on specialized units at a given site while pushing to accommodate volatile requirements. They take evolving market preferences into consideration, up to a feasible extent	Software teams work in a self-contained mode to align with their specific work specializations for a given site. They rely sporadically on their counterparts, to manage regular client expectations in a committed manner
Inter-linked	Software teams are pushed to work in a collaborative manner on their specialization at a given site, while faced with handling extensive mutual dependencies across sites. They must consider evolving market preferences, up to a feasible extent	Software teams push to work in a collaborative manner on their specialization at a given site, while faced with frequent cross-site coordination of tasks. This helps them manage client expectations in a committed manner

P2: Given an ADD setting, the type of distributed team configuration—whether it is autonomous or inter-linked—has a direct bearing on the response to the need for stability

Integrating the pieces of evidence gathered across our case settings, Table 7 summarizes the findings on the notion of conflict between flexibility and stability for the ADD setting. This presents a nuanced perspective on the central conflict of an ADD team's need to accommodate change requests versus ensuring that the project progresses as per plan.

5.3 Managing the conflict between flexibility and stability

The dual demands of flexibility and stability across our case settings have teams grappling with local, site-related work responsibilities while coordinating global, cross-site dependencies. Our investigation reveals that performance elements (e.g., clarity in role specification, time management of meetings) enable teams to manage their responsibilities effectively, whereas social elements (e.g., type of boundary spanning, extent of mutual responsiveness) help them handle cross-site relations.

We observe different kinds of managerial patterns across the cases (see Table 8), which provides us guidance for conflict management in an ADD setting.

Upon closer investigation, we witness the interplay between specific performance and social elements, which results in innovative coping mechanisms for the ADD setting. We choose to illustrate only those that prominently manifest in the projects that we have considered.

5.3.1 Interplay between Clarity in role specification and Type of boundary spanning

Operating in an autonomous distributed configuration, *Proj ChipSys* relies on sharply defined responsibilities for each team member. A dedicated manager role is responsible for tracking the progression of the project work. An execution lead ensures that all of the work from the India site is handled in an adequate manner. Consequently, the setup is able to establish clear standards of performance and behaviour. The key responsibilities are clearly stated, and individuals take efforts to ensure that they fulfil them. This enables the teams to work in a self-contained mode and to handle their commitments in a responsive manner (*stability*).

With such clearly defined roles, we have seen that *Proj ChipSys* accords the freedom to team members to span cross-site boundaries and reach out to their counterparts to discuss and resolve project related matters. We also observe that clearly defining the responsibilities for each team member results in specific ownership of project matters across sites. For instance, the managers collaborate across sites to discuss the progression of the overall project. The execution lead reaches out to their counterparts to provide intra-team updates and to seek clarifications.

A similar scenario can be witnessed in the autonomous context of *Proj HotelSys*. Except for the iteration manager (who also plays the Scrum Master role), the setting clearly defines roles for a business analyst, quality assurance analyst and developer. Again, team members have the freedom to reach out to their counterparts to conduct role-relevant discussions. This facilitates ease of handling cross-site relations especially in the wake of regularly evolving requirements (*flexibility*).

In an inter-linked distributed configuration, we witness role specifications and boundary spanning decisions that are distinct from the autonomous setting. *Proj HealthSys* relies on a significant overlap in the role definition of team members with the program-level individuals; responsibilities intersect across the engineering manager, Scrum Master, release train engineer and value stream engineer roles. These overlaps enable the teams to allow multiple individuals to keep a focus on key project management activities, such as tracking work progression for a given site. Subsequently, they prefer to work in a dependent mode and handle their responsibilities in a collaborative manner *(stability)*.

Tab. 8 Managerial patterns for ADD setting

Type of ADD project investigated / *Project Context*		#1 [Product Development – Autonomous]		#2 [Product Development – Inter-linked]		#3 [Services – Autonomous]		#4 [Services – Inter-linked]	
Performance Elements	Clarity in Role Specification	*High*	*Low*	*High*	*Low*	*High*	*Low*	*High*	*Low*
	Time Management of Meetings	*Precise*	*Flexible*	*Precise*	*Flexible*	*Precise*	*Flexible*	*Precise*	*Flexible*
	Discretionary Work Effort	*Independent Team Members*	*Shared across Teams*	*Independent Team Members*	*Shared across Teams*	*Independent Team Members*	*Shared across Teams*	*Independent Team Members*	*Shared across Teams*
	Ownership of Code	*Weak Individual*	*Collective*	*Weak Individual*	*Collective*	*Weak Individual*	*Collective*	*Weak Individual*	*Collective*
Social Elements	Type of Boundary Spanning	*Designated Individual*	*Entire Team*	*Designated Individual*	*Entire Team*	*Designated Individual*	*Entire Team*	*Designated Individual*	*Entire Team*
	Extent of Mutual Responsiveness	*Dedicated Representation*	*Focused Intervention*	*Dedicated Representation*	*Focused Intervention*	*Dedicated Representation*	*Focused Intervention*	*Dedicated Representation*	*Focused Intervention*
	Visibility of Work Progression	*Partial*	*Complete*	*Partial*	*Complete*	*Partial*	*Complete*	*Partial*	*Complete*
	Team Inclusivity in Meetings	*Relevant Team Members*	*Entire Team*	*Relevant Team Members*	*Entire Team*	*Relevant Team Members*	*Entire Team*	*Relevant Team Members*	*Entire Team*

Each of the grey colored boxes denote the synthesized evidence (across two projects for a given ADD type) gathered from the field

In the presence of such overlaps, we see an attempt to manage the chaos that arises from the shared responsibilities. The setting relies on a designated cross-site boundary spanner, in the form of a dedicated release train engineer who resolves project related matters. This facilitates dedicated cross-site coordination of tasks in the wake of updated customer demands (*flexibility*).

Overall, the clarity in role specification drives the cross-site boundary spanning choices across three of the four projects under study. The remaining project serves to highlight an exception. *Proj TrainSys* (with inter-linked configuration) functions with an overlap in role specifications across manager, feature owner, business analyst and technical architect. Despite such role overlaps, their team members can partake in cross-site discussions. Consequently, the setup engages in extensive intra-site discussions to ensure that miscommunication is avoided. However, the teams still experience chaos in the setting owing to multiple voices that generate a communication trap on project relevant matters.

In light of this evidence, we propose the following:

P3a: For an autonomous ADD configuration, if there is clarity in role specification, then the flexibility-stability conflict is best managed when the entire team is given the freedom to span cross-site boundaries for project related matters.

P3b: For an inter-linked ADD configuration, if there is an overlap in role specification, then the flexibility-stability conflict is best managed when a designated individual is assigned to span cross-site boundaries for project related matters.

5.3.2 Interplay between Ownership of code and Team inclusivity in meetings

Both product development cases (*Proj ChipSys* and *Proj HealthSys*) have embraced weak ownership of code modules; any team member can modify the code, provided the module's owner is kept in the loop. The accountability for a particular code module lies with a specific developer, which ensures that each team member actively handles their respective modules. At an aggregate level, this enables team commitments to be managed responsively (*stability*).

With the authority being split across individuals for separate code modules, we have observed that the entire team participates in formal meetings, such as sprint planning and reviews. The planning meeting enables independent code owners to understand the expectations of the module. The review meeting requires them to showcase developed modules and receive feedback on the functionality. Even regular ceremonies, such as daily stand-up meetings mandate participation by all team members to share updates on the work progression. Such an elevated level of inclusivity ensures that the entire team discusses project related matters among themselves (in regular meetings) as well as in the presence of counterparts (in formal meetings) on an ongoing basis. This facilitates the ease of handling project contingencies, especially during instances of updated market preferences (*flexibility*).

In direct contrast, both services-oriented cases (*Proj HotelSys* and *Proj TrainSys*) rely on collective code ownership, wherein the whole team takes accountability for the entire codebase. Whenever a specific issue is raised over

a module, the teams do not attribute it to a specific developer. Instead, the entire team readily takes account of the issue and seeks to jointly address it. This approach enables the entire team to appreciate specific features related to the codebase. They collaborate to ensure that bugs are resolved on an immediate basis, creating a strong focus to fulfil team-level commitments (*stability*).

In the presence of collective ownership, we have observed that a subset of team members may participate in formal meetings such as sprint planning and review. With *Proj HotelSys* and *Proj TrainSys*, we have observed a quality assurance analyst taking the lead and presenting the software artefact to the customer at the review meeting. Along similar lines, a regular meeting, such as a Scrum of Scrums, does not necessitate participation from the entire team. In most of the cases, the Scrum Master and/or quality assurance personnel take the responsibility to present and discuss team-level progression on project work with their counterparts.

Whenever customer demands must be refined, these representatives take the lead and engage in relevant discussions within their team, appraising the members of changes to the scope of requirements, and preparing to embrace modifications to the backlog. Subsequently, these individuals interface with their counterparts to discuss dependencies or intermittent reliance on each other's team for completion of the tasks. Such an approach enables the committed handling of cross-site efforts on project related matters (*flexibility*).

Overall, the code ownership approach drives team member inclusivity in meetings across all the projects under study. Considering this evidence, we propose the following:

P4a: For a product-focused ADD setting, if there is collective ownership of code, then the flexibility-stability conflict is best managed when the relevant team members actively participate in meetings.
P4b: For a services-oriented ADD setting, if there is weak (individual) code ownership, then the flexibility-stability conflict is best managed when the entire team actively participates in meetings.

5.3.3 Interplay between Time management of meetings and Visibility of work progression

Given an autonomous distributed configuration, *Proj ChipSys* and *Proj HotelSys* rely upon complete visibility of work progression, in terms of the status of stories, task dependencies and trends. They have adopted a proprietary tool that consists of virtual storyboard, which is readily accessible by each project stakeholder. The teams regularly update their progress on the tool to ensure that the most recent status of stories (completed, in-progress or yet to be taken up) is made available across sites. Every individual can track the movement of stories at any given point in time. They also mark dependent stories on the board to clearly convey target dates for handling such work across sites. The tool also depicts the burndown chart for a given site and metrics such as velocity. These artefacts facilitate teams in managing cross-site discussions on project related matters, especially in the wake of regularly evolving requirements (*flexibility*).

When teams maintain complete visibility of work progression, we have seen that they conduct meetings in a disciplined manner. Each of the two case projects follows daily stand-up for a fixed duration, wherein the team members refer to the storyboard and update individual progress. Along similar lines, meetings such as Scrum of Scrums or their derivatives (e.g. XFT meeting in the case of *Proj ChipSys)* are also held for a precise time duration; the teams are reasonably aware of the discussion points and take stock of dependencies by referring to the virtual board. Formal meetings such as planning and review are also held in a timely manner. While team members discuss concerns and voice their opinions on stories, the meetings are completed on time. Such a disciplined nature of conducting meetings ensure that teams remain focused on their commitments and strive towards fulfilling them *(stability)*.

A contrasting picture emerges for an inter-linked distributed configuration. Both the case projects (*Proj HealthSys* and *Proj TrainSys*) must rely on partial visibility of work progression. Despite sharing the most recent status of stories and flagging the movement on dependent tasks with a proprietary tool, we notice that *Proj HealthSys* teams refrain from sharing site-relevant trends (such as velocity and burndown charts) with their counterparts; they prefer to keep such metrics internal to a given site. Members of *Proj TrainSys* do not tag cross-site dependencies for reference. Instead, they prefer to discuss them over regular meetings. Overall, such partial visibility results in teams grappling with clarity on project movement from across sites on a real-time basis. When change requests arrive, this pushes them to engage in task-directed discussions with counterparts to handle cross-site dependencies in a responsive manner (*flexibility*).

When teams maintain partial visibility of work progression, we observe that they conduct meetings in a more flexible manner. *Proj HealthSys* presents evidence for team members stretching beyond the set agenda for regular meetings. Apart from sharing updates, the team members also discuss impediments faced in their tasks in daily stand-up meetings. Consequently, the teams end up creating bandwidth beyond the stipulated time to ensure such discussions could be accommodated. We witness a similar trend in planning as well as review meetings. The planning meeting involves detailed discussion on the stories across sites, whereas the review meeting entails rich feedback on the developed stories. In addition, *Proj TrainSys* engages in the conduct of meetings such as story kick-off and desk check, which are respectively held at the beginning and completion of

each story. The objective of these meetings is to discuss the undertaken story for development in detail and to ensure that the acceptance criteria are being followed. Consequently, the teams follow a nimble approach towards the conduct of such meetings. This enables teams to gain significant clarity (through planning and kick-off) or process feedback (through review and desk check) on project deliverables to ensure that site-level commitments are seldom compromised *(stability)*.

Overall, the visibility of work progression drives the time management of meetings across all the case projects under study. In light of this evidence, we propose the following:

P5a: For an autonomous ADD configuration, if there is complete visibility of work progression, the flexibility-stability conflict is best managed when formal as well as regular meetings are conducted in a time-precise manner.
P5b: For an inter-linked ADD configuration, if there is partial visibility of work progression, the flexibility-stability conflict is best managed when formal as well as regular meetings are conducted in a time-flexible manner.

5.3.4 Interplay between Discretionary work effort and Mutual responsiveness across sites

Given an autonomous configuration, software teams from both the case projects (i.e. *Proj ChipSys* and *Proj HotelSys*) work in a self-contained mode at a given site. Each of the team handles either a specific module, feature or a component. For instance, *Proj ChipSys* involves display/graphic module being handled at India site, the camera module getting developed at the Germany site, and the video module being managed out of the China site. In such a scenario, we witness independent discretionary work effort of team members at a given site. They rely upon extensive intra-team support, where team individuals assume additional responsibilities, to ensure that the module is handled adequately. *Proj ChipSys* entails swiftness of the team members to seek immediate clarifications among themselves and proceed with development at a given site. *Proj HotelSys* follows the concept of core and feature team members, who regularly stretch their work boundaries. For instance, the technical lead (core team member) responsible for addressing code-related concerns, also takes up stories for development to assist feature team members. This high level of intra-team support enables each of the team to handle their commitments such that project objectives are adequately met *(stability)*.

In the presence of such intra-team discretionary work effort, we observe that the teams rely upon the focused intervention of specific members from across sites for project related queries. As the teams prefer to work in a self-contained mode, they seem reluctant as a group to provide additional priority to sort cross-site related issues. *Proj ChipSys* presents evidence of teams across site acknowledging each other's concern yet seldom initiating prompt actions. Thus, the teams rely upon repeated checks with specific individuals (i.e., Product Owner in these cases) to receive adequate responses. *Proj HotelSys* entails focused intervention from onsite team members that act as mediators between offshore teams and client as well as other vendor teams. In this scenario, the onsite team takes up the responsibility to assist offshore teams in their endeavour to coordinate discussions across the site. This enables the teams to conduct focused cross-site coordination for project relevant matters, especially during instances of updated customer expectations (*flexibility*).

On the other hand, our inter-linked distributed configuration cases (*Proj HealthSys* and *Proj TrainSys*) involve software teams working in a collaborative manner at a given site. *Proj HealthSys* relies upon the willingness of teams to lend additional support to each other for handling contingencies faced at a given site. Similarly, *Proj TrainSys* depends upon leveraging the expertise of members across teams. Especially during instances when an individual has spent considerable time at the distributed site, he/she is willing to readily share relevant insights across the teams. This high level of inter-team support enables the teams to jointly manage their commitments such that the dependent work is handled in an effective manner *(stability)*.

In the presence of such shared discretionary work effort between teams at a given site, we witness that the teams rely upon dedicated representation from across sites for project related queries. As the teams engage with their demanding work effort for a given site, they seem to place the emphasis on a particular individual to ensure that cross-site coordination needs are sorted. *Proj HealthSys* relies upon the effort of a release train engineer to coordinate with the counterpart on project related matters. *Proj TrainSys* relies upon a specific representative who sits with the distributed unit to ensure that dependencies are effectively handled. This facilitates dedicated cross-site coordination of dependencies in the wake of an updated scope of requirements (*flexibility*).

Overall, the kind of discretionary work effort followed at a given site drives the nature of mutual responsiveness with the counterparts, across all the case projects under study. In light of this evidence, we propose the following:

P6a: For an autonomous ADD configuration, if there is independent discretionary work effort of team members, the flexibility-stability conflict is best managed when each team relies upon focused intervention of certain individuals for mutual responsiveness across sites.
P6b: For an inter-linked ADD configuration, if there is shared discretionary work effort between teams, the flexibility-stability conflict is best managed when the cohort of teams rely upon dedicated representation of an individual for mutual responsiveness across sites.

To summarize, each of the four interactions presented here involves a specific *performance element* and a *social element* (cf. [28]. The software teams across case projects have devised focused strategies to synchronize the handling of work responsibilities and cross-site relations. Thus, the project context serves to enable the software teams to execute their respective ADD efforts in a smooth manner.

Based on the findings from across the case projects, we present our ADD management framework (see Fig. 2) as follows.

6 Discussion

A major contribution of our study entails providing greater clarity on the flexibility-stability conflict in a wide variety of ADD settings. In addition, this research extends ambidexterity literature by explaining the interplay between the performance and social elements for a given project context, by suggesting concrete steps on how software teams can manage these forces, based on empirical evidence.

6.1 Theoretical implications

Previous studies that have examined trade-offs in managing projects largely focus on the exploration–exploitation dilemma at an organization-level. In the IS domain, research has explicitly considered the trade-off between alignment-adaptability at the business unit or team level; we harness this further to characterize the flexibility-stability conflict. Alignment refers to the pursuit of the entire unit working towards a common goal, whereas adaptability concerns the unit's ability to meet changing demands in the task environment [28]. Researchers [9, 78] have used the flexibility-stability construct to highlight inherent conflicting forces in a given setting.

Lee et al. [45] emphasizes that agile methods, which encourage teams to embrace change, must be modified to incorporate more rigor and discipline in globally distributed contexts. Ramesh et al. [64, 65] highlight competing demands between the tenets of agile and distributedness in ADD projects; whereas an agile setting relies on informal processes to facilitate coordination, distributed settings typically employ formal mechanisms. Our study provides insights beyond this conceptualization of the central conflict in ADD settings. Our research identifies contextual elements that help us characterize the dynamics of these conflicting forces.

This study has implications for the theory of ambidexterity, which has been recognized as a valuable approach to understanding the dynamics of contingencies faced by organizations, business units and/or teams. Gibson and Birkinshaw [28] have characterized the elements of contextual ambidexterity, which guides business units and teams to handle conflicting forces.

Existing research [45, 64, 65] suggest techniques that teams can adopt to mitigate the conflict. Lee et al. [45] characterizes these coping strategies along two dimensions: (1) Whether the strategy concerns the initiation or execution phase of the project lifecycle, and (2) Whether the strategy focuses on task-related, people-related or technology-related aspects. Ramesh et al. [64, 65] suggest practices in the form of formal and informal processes, which give rise to specific mitigation strategies to the conflicting demands. They map these strategies to the well-known antecedents (discipline, stretch, support and trust) of contextual ambidexterity.

Our work builds upon the approaches of the predecessors and specializes it further. Rather than directly focusing on general strategies (cf. [28] adopted by software teams, we leverage the notion of project context in each ADD setting to derive insights. We explicate *discipline* aspects in terms of (1) clarity in role specifications and (2) time management of meetings, and *stretch* aspects such as (3) discretionary work effort and (4) ownership of code; *support* elements in terms of (5) type of boundary spanning and (6) extent of mutual responsiveness; *trust* established by (7) visibility of work progression and (8) team member inclusivity in meetings. We provide exemplars for how these elements manifest themselves in practice. Our case settings reveal interactions between performance and social elements in each ADD context, which help software teams manage the conflicting forces that are present in the setting.

6.2 Practical implications

The domain of software development is characterized by a high level of requirements uncertainty [11], complexities of coordination arising from task inter-dependencies, artefact evolution through continuous client involvement, wide swings in task performance by personnel with similar backgrounds [20], and so on. General directives of organization theory do not directly translate to this context, given these differences. By consulting our case studies, practitioners specifically executing ADD projects can develop clarity on the nature of conflict between flexibility and stability that are prevalent in their setting. This research explores these forces in detail and provides a nuanced understanding of different types of ADD setups. Contingent upon their context (see Fig. 1), the managers would be able to characterize the fundamental conflict in their setting. Equipped with this understanding, they can better execute their project implementation.

Moreover, this study shall assist practitioners in future ADD implementations by providing guidance on conflict management for the setting. This becomes even more relevant given the current pandemic scenario which has made

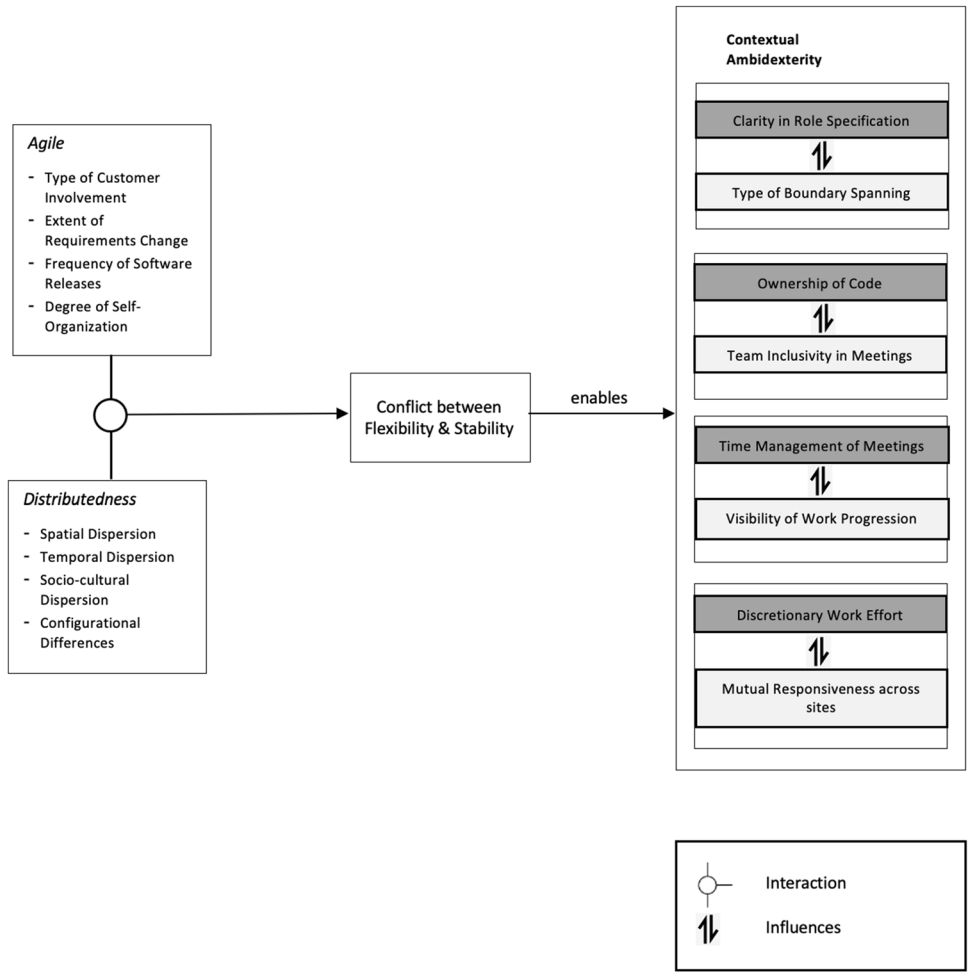

Fig. 2 ADD Managerial Framework

distributed mode of work prominent. An empirical contribution of this study is the ADD managerial framework (see Fig. 2) that we propose for handling such efforts. Through multiple case studies with varied agile and distributedness characteristics, we present the framework in action across several scenarios. The contextual elements reveal specific managerial patterns (see Table 8) for a given setting. These patterns could be leveraged in practice to develop team-level strategies for effective handling of the conflicting forces.

Subsequently, this study serves to inform managers to diagnose problems that may be present in an existing context. The interactions between the specific performance and social elements highlighted in this research may be compared with the prevalent contextual elements. In the event of a mismatch, inferences could be drawn from this study to make the project context more conducive to handle flexibility-stability conflict present in the setting.

This study also provides pointers to individuals on an ADD team. Our research reveals that the approach adopted by teams towards code ownership drives member participation in meetings. Moreover, clarity in role specifications has an influence on cross-site boundary spanning decisions. Such findings have implications for the salience of team dynamics in a given ADD setup.

Finally, our in-depth case studies paint a valuable picture of different types of ADD setups and lay a foundation for future researchers to develop and test propositions that are presented in this research.

7 Conclusion

Since early 2000s, agile has primarily been considered suitable for a collocated work setting. However, the global realities continued the push for adapting agile in a distributed context. Fast forward to the 2020s, the pandemic has prompted organizations to foster hybrid work models; the restriction of collocation for agile teams has been further relaxed. ADD studies have gained more prominence in the wake of this development. Our research focuses on a central issue of analyzing inherent conflicts and their management in varied ADD setups.

The findings reveal that product-focused agile cases tend to implement only feasible updates to the market

demands, whereas the services-oriented cases exhibit a more committed handling of modifications arising from changed client expectations.

In our studies, we have seen one or more software teams operate out of a single geography. We find that the autonomous configuration forces each of these teams to work in a self-contained mode from their respective project site, whereas the inter-linked configuration results in these teams working in a collaborative mode to manage their commitments.

In summary, our research reveals that the type of agile project engagement, i.e., product development versus services, drives the response to demand for *flexibility*, whereas the distributed team configuration, viz. autonomous versus inter-linked split, drives the response to the need for *stability* in the ADD setting. To manage the conflicting forces, our case settings reveal interactions between performance and social elements in each ADD context. By consulting our cases, managerial patterns could be leveraged in practice to develop relevant strategies for effective handling of the conflicting forces.

Funding No funding was received for conducting this study.

Declarations

Conflict of interests The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Agile alliance. (2001). Retrieved from http://agilemanifesto.org/ (Accessed on 1st June 2021)
2. Alzoubi YI, & Gill AQ (2014). Agile global software development communication challenges: a systematic review. Paific Asia Conference on Information Systems (PACIS)
3. Arora A, Arunachalam VS, Asundi J, Fernandes R (2001) The Indian software services industry. Res Policy 30(8):1267–1287
4. Baheti P, Gehringer E, Stotts D (2002) Exploring the efficacy of distributed pair programming. Extreme programming and agile methods—XP/Agile Universe 2002. Springer, Berlin Heidelberg, pp 208–220
5. Baldwin CY, Clark KB (2003) Managing in an age of modularity. Manag Modular Age: Archit, Netw, Organ 149:84–93
6. Benbasat I, Goldstein DK, Mead M (1987) The case research strategy in studies of information systems. MIS Q 11(3):369–386
7. Benner MJ, Tushman ML (2015) Reflections on the 2013 decade award - "exploitation, exploration, and process management: the productivity dilemma revisited" ten years later. Acad Manag Rev 40(4):497–514
8. Benner MJ, Tushman ML (2003) Exploitation, exploration, and process management: the productivity dilemma revisited. Acad Manag Rev 28(2):238–256
9. Boehm B, Turner R (2004). Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods. In Software Engineering, 2004. ICSE 2004. In: Proceedings. 26th International Conference on (pp. 718–719). IEEE.
10. Braithwaite K, Joyce T (2005) XP expanded: distributed extreme programming. Extreme programming and agile processes in software engineering. Springer, Berlin Heidelberg, pp 180–188
11. Brooks FP (1987) No silver bullet. IEEE. Computer 20(4):10–19
12. Cao L, Mohan K, Ramesh B, Sarkar S (2013) Evolution of governance: achieving ambidexterity in IT outsourcing. J Manag Inf Syst 30(3):115–140
13. Carlshamre P, Sandahl K, Lindvall M, Regnell B, Nattoch Dag J (2001). An industrial survey of requirements interdependencies in software product release planning. In: Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, Los Alamitos, CA, IEEE
14. Carlshamre P (2002) Release planning in market driven software product development: provoking an understanding. Requir Eng 7(3):139–151
15. Cataldo M (2007). Dependencies in geographically distributed software development: overcoming the limits of modularity. Doctoral dissertation, Carnegie Mellon University
16. Chow T, Cao DB (2008) A survey study of critical success factors in agile software projects. J Syst Softw 81(6):961–971
17. Cockburn A (2002) Agile software development. Pearson Education, Boston
18. Corbin J, Strauss A (2008) Basics of qualitative research: techniques and procedures for developing grounded theory. California, CA, USA, Thousand Oaks
19. Dahlstedt A, Karlsson L, Persson A, NattochDag J, Regnell B (2003). Market-driven requirements engineering processes for software products – a report on current practices. International Workshop on COTS and Product Software RECOTS, held in conjunction with the 11th IEEE International Requirements Engineering Conference, Los Alamitos, CA, IEEE
20. DeMarco T, Lister T (2013) Peopleware: productive projects and teams. Addison-Wesley, Heidelberg
21. Dorairaj S, Noble J, Malik P (2011) Effective communication in distributed Agile software development teams. In: Sillitti A, Hazzan O, Bache E, Albaladejo X (eds) Agile processes in software engineering and extreme programming. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 102–116. https://doi.org/10.1007/978-3-642-20677-1_8
22. Dubé L, Paré G (2003) Rigor in information systems positivist case research: current practices, trends, and recommendations. MIS Q 27(4):597–636. https://doi.org/10.2307/30036550
23. Dullemond K, van Gameren B, Van Solingen R (2009). How technological support can enable advantages of agile software development in a GSE setting. In: Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on (pp. 143–152). IEEE
24. Duncan RB (1976) The ambidextrous organization: designing dual structures for innovation. Manag Org 1:167–188
25. Eisenhardt KM (1989) Building theories from case study research. Acad Manag Rev 14(4):532–550
26. Fogelström ND, Gorschek T, Svahnberg M, Olsson P (2010) The impact of agile principles on market-driven software product development. J Softw Maint Evol Res Pract 22(1):53–80
27. Ghoshal S, Bartlett CA (1994) Linking organizational context and managerial action: the dimensions of quality of management. Strateg Manag J 15(S2):91–112
28. Gibson CB, Birkinshaw J (2004) The antecedents, consequences, and mediating role of organizational ambidexterity. Acad Manag J 47(2):209–226
29. Glaser BG (1978). Advances in the methodology of grounded theory: theoretical sensitivity

30. Glaser BG (1965) The constant comparative method of qualitative analysis. Soc Probl 12(4):436–445
31. Greer D, Ruhe G (2004) Software release planning: an evolutionary and iterative approach. Inf Softw Technol 46(4):243–253
32. Gregory RW, Keil M, Muntermann J, Mähring M (2015) Paradoxes and the nature of ambidexterity in IT transformation programs. Inf Syst Res 26(1):57–80
33. Hoda R, Noble J, Marshall S (2011) The impact of inadequate customer collaboration on self-organizing Agile teams. Inf Softw Technol 53(5):521–534
34. Hole S, Moe NB (2008) A case study of coordination in distributed agile software development. Software process improvement. Springer, Berlin Heidelberg, pp 189–200
35. Holmström H, Fitzgerald B, Ågerfalk PJ, Conchúir EÓ (2006) Agile practices reduce distance in global software development. Inf Syst Manag 23(3):7–18
36. Huang J, Newell S, Huang J, Pan SL (2014) Site-shifting as the source of ambidexterity: empirical insights from the field of ticketing. J Strateg Inf Syst 23(1):29–44
37. Hummel M, Rosenkranz C (2014). Measuring the impact of communication in agile development: a research model and pilot test. In: Proceedings of the Nineteenth Americas Conference on Information Systems
38. Jalali S, Wohlin C (2012) Global software engineering and agile practices: a systematic review. J Soft: Evolut Process 24(6):643–659
39. Jensen B, Zilmer A (2003) Cross-continent development using Scrum and XP. In: Marchesi M, Succi G (eds) Extreme Programming and Agile Processes in Software Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 146–153. https://doi.org/10.1007/3-540-44870-5_19
40. Kietzmann J, Plangger K, Eaton B, Heilgenberg K, Pitt L, Berthon P (2013) Mobility at work: a typology of mobile communities of practice and contextual ambidexterity. J Strateg Inf Syst 22(4):282–297
41. Kircher M, Jain P, Corsaro A, Levine D (2001) Distributed extreme programming. Extreme Programming and Flexible Processes in Software Engineering, Italy, pp 66–71
42. Korkala M, Abrahamsson P (2007) Communication in distributed agile development: a case study. In: Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on (203–210). IEEE
43. Korkala M, Abrahamsson P, Kyllonen P (2006) A case study on the impact of customer communication on defects in agile software development. In Agile Conference, 2006, IEEE
44. Lee S, Yong HS (2010) Distributed agile: project management in a global environment. Empir Softw Eng 15(2):204–217
45. Lee G, DeLone W, Espinosa JA (2006) Ambidextrous coping strategies in globally distributed software development projects. Commun ACM 49(10):35–40
46. Lehtola L, Kauppinen M (2006) Suitability of requirements prioritization methods for market-driven software product development. Softw Process: Improv Pract 11(1):7–19
47. Levinthal DA, March JG (1993) The myopia of learning. Strateg Manag J 14(S2):95–112
48. Malone TW, Crowston K (1994) The interdisciplinary study of coordination. ACM Comput Surv (CSUR) 26(1):87–119
49. Maruping LM, Venkatesh V, Agarwal R (2009) A control theory perspective on agile methodology use and changing user requirements. Inf Syst Res 20(3):377–399
50. Melnik G, Maurer F (2004). Direct verbal communication as a catalyst of agile knowledge sharing. In Agile Development Conference, 2004. IEEE
51. Miles MB, Huberman AM (1994) Qualitative data analysis: an expanded sourcebook. Sage
52. Nambisan S (2001) Why service business are not product businesses. MIT Sloan Manag Rev 42(4):72
53. Napier NP, Mathiassen L, Robey D (2011) Building contextual ambidexterity in a software company to improve firm-level coordination. Euro J Inform Sys 20(6):674–690
54. Nosella A, Cantarello S, Filippini R (2012) The intellectual structure of organizational ambidexterity: a bibliographic investigation into the state of the art. Strateg Organ 10(4):450–465
55. Reilly CA, Tushman ML (2004) The ambidextrous organization. Harv Bus Rev 82(4):74–83
56. O'Leary MB, Cummings JN (2007) The spatial, temporal, and configurational characteristics of geographic dispersion in teams. MIS Q 31(3):433–452
57. Paasivaara M, Lassenius C, Heikkila VT (2012). Inter-team coordination in large-scale globally distributed scrum: do Scrum-of-Scrums really work? In: Empirical Software Engineering and Measurement (ESEM), 2012 ACM-IEEE International Symposium on (pp. 235–238). IEEE
58. Parnas DL (2001) On the criteria to be used in decomposing systems into modules. In: Broy M, Denert E (eds) Pioneers and their contributions to software engineering. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 479–498. https://doi.org/10.1007/978-3-642-48354-7_20
59. Patel PC, Messersmith JG, Lepak DP (2013) Walking the tightrope: an assessment of the relationship between high-performance work systems and organizational ambidexterity. Acad Manag J 56(5):1420–1442
60. Persson JS, Mathiassen L, Aaen I (2012) Agile distributed software development: enacting control through media and context. Inform Sys J 22(6):411–433
61. Podgurski A, Clarke LA (1990) A formal model of program dependences and its implications for software testing, debugging, and maintenance. IEEE Trans Softw Eng 16(9):965–979
62. Potts C (1995) Invented requirements and imagined customers: requirements engineering for off-the-shelf software. In: Requirements Engineering, 1995, Proceedings of the Second IEEE International Symposium on (pp. 128–130). IEEE
63. Raisch S, Birkinshaw J, Probst G, Tushman ML (2009) Organizational ambidexterity: balancing exploitation and exploration for sustained performance. Organ Sci 20(4):685–695
64. Ramesh B, Cao L, Mohan K, Xu P (2006) Can distributed software development be agile? Commun ACM 49(10):41–46
65. Ramesh B, Mohan K, Cao L (2012) Ambidexterity in agile distributed development: an empirical investigation. Inf Syst Res 23(2):323–339
66. Regnell B, Brinkkemper S (2005) Market-driven requirements engineering for software products. Engineering and managing software requirements. Springer, Berlin, Heidelberg, pp 287–308
67. Saldaña J (2014) Coding and analysis strategies. In The Oxford handbook of qualitative research, Heidelberg
68. Schulze P, Heinemann F, Abedin A (2008). Balancing exploitation and exploration. In: Academy of Management Proceedings (Vol. 2008, No. 1, pp. 1–6). Academy of Management 2008
69. Simsek Z, Heavey C, Veiga JF, Souder D (2009) A typology for aligning organizational ambidexterity's conceptualizations, antecedents, and outcomes. J Manag Stud 46(5):864–894
70. Šmite D, Moe NB, Ågerfalk PJ (2010) Fundamentals of agile distributed software development. Agility across time and space. Springer, Berlin Heidelberg, pp 3–7
71. Staats BR, Brunner DJ, Upton DM (2011) Lean principles, learning, and knowledge work: evidence from a software services provider. J Oper Manag 29(5):376–390
72. Sutherland J, Viktorov A, Blount J, Puntikov N (2007) Distributed scrum: agile project management with outsourced development teams. In: System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on pp. 274a-274a. IEEE

73. Tiwana A (2008) Do bridging ties complement strong ties? An empirical examination of alliance ambidexterity. Strateg Manag J 29(3):251
74. Tiwana A (2010) Systems development ambidexterity: explaining the complementary and substitutive roles of formal and informal controls. J Manag Inf Syst 27(2):87–126
75. Tushman ML, O'Reilly CA III (1996) Managing evolutionary and revolutionary change. Calif Manag Rev 38(4):8–28
76. Van Looy B, Martens T, Debackere K (2005) Organizing for continuous innovation: on the sustainability of ambidextrous organizations. Creat Innov Manag 14(3):208–221
77. Vanhaverbeke W, Peeters N (2005) Embracing innovation as strategy: corporate venturing, competence building and corporate strategy making. Creat Innov Manag 14(3):246–257
78. Vinekar V, Slinkman CW, Nerur S (2006) Can agile and traditional systems development approaches coexist? Ambidextrous View Inf Syst Manag 23(3):31–42
79. Xu L, Brinkkemper S (2007) Concepts of product software. Eur J Inf Syst 16(5):531–541
80. Yin RK (2009). Case study research: design and methods, 5. Sage
81. Yin RK (2003) Case study research - design and methods. Applied social research methods series, 5. Sage
82. Zimmermann A, Raisch S, Cardinal LB (2018) Managing persistent tensions on the frontline: a configurational perspective on ambidexterity. J Manag Stud 55(5):739–769