International Journal of Information Management 44 (2019) 13–24

Contents lists available at ScienceDirect

International Journal of Information Management

journal homepage: www.elsevier.com/locate/ijinfomgt

Relationships between IT department culture and agile software development practices: An empirical investigation

Manjul Gupta[a,*], Joey F. George[b], Weidong Xia[a]

[a] Department of Information Systems and Business Analytics, College of Business, Florida International University, Miami, FL 33199, United States
[b] Department of Supply Chain & Information Systems, College of Business, Iowa State University, 3311 Gerdin Business Building, Ames, IA 50011, United States

ARTICLE INFO

Keywords:
Agile practices
Organizational culture
Competing values model
IT department

ABSTRACT

IT department culture has been widely recognized as an important factor that influences the adoption of agile practices. Yet, the research pertaining to the relationship between IT department culture and agile practices usage remains underexplored. This study proposes and tests the relationships between four competing cultural forms and two types of agile practices - social and technical. The findings contribute to the extant literature by integrating the competing values model of culture into the literature on factors affecting agile development at the IT department level.

1. Introduction

As business and technology environments become increasingly uncertain and dynamic, agile software development (ASD) has been adopted by information technology (IT) departments in various organizations as a response to the failure of traditional plan-driven waterfall-based approach (Berger, 2007; Larson & Chang, 2016). ASD encompasses agile values and principles, agile methods, and agile practices (Conboy, 2009; Dingsøyr, Nerur, Balijepally, & Moe, 2012). As described by the *Agile Manifesto*, agile values and principles prescribe the overarching philosophies and guidelines underlying ASD. ASD methods such as eXtreme Programming (XP) (Beck, 1999) and Scrum (Schwaber & Beedle, 2002) have emerged as methodological frameworks that organizations can adopt to materialize agile values and principles. ASD methods are composed of various practices (Tripp & Armstrong, 2014). In this paper, we focus on the two types of ASD practices and then investigate how different forms of IT department cultures affect their implementation.

Like the traditional plan driven waterfall-based software development approach, ASD encompasses several technical procedures; however, ASD distinguishes itself from the waterfall approach by being people-centric with drastic emphasis on fast-cycled iterative social interactions between and among different individuals involved in the software development process (Dybâ & Dingsoyr, 2009; Hung, Hsu, Su, & Huang, 2014). ASD "adheres to the concept of software development as a continuous and repetitive social and technical engagement and the need to establish daily routines that gradually generate pieces of functional software. These daily and weekly routines rely on multiple technical and social" practices (Thummadi, Shiv, Berente, & Lyytinen, 2011;). A number of scholars have characterized different ASD practices into technical and social groups (Asnawi, Gravell, & Wills, 2011; Chow & Cao, 2008; McHugh, Conboy, & Lang, 2011; Mnkandla & Dwolatzky, 2007; Robinson & Sharp, 2005b; Treude & Storey, 2009). Drawing on this body of literature, Hummel, Rosenkranz, and Holten, (2015) conducted a qualitative study and proposed a new construct called "*social agile practices*." They define the social agile practices construct as the ASD practices that facilitate social interaction, collaboration, and direct communication, whereas the t*echnical agile practices* construct refers to the coding/testing-oriented software engineering practices. In this study, we rely on Hummel et al. (2015) categorization of ASD practices into social and technical types. Moreover, ASD is considered a highly socio-technical process, and thus, it is critical to carefully include both agile-oriented social and technical practices to understand the impact of ASD in detail (Bellini, Pereira, & Becker, 2008; Whitworth & Biddle, 2007).

Additionally, past research has identified IT department culture as a significant barrier to the adoption of agile practices (Cao, Mohan, Xu, & Ramesh, 2009; Fruhling & Tarrell, 2008). Consequently, substantial research has concentrated on proposing the ideal agile culture, which tends to be people-centered and collaborative (Cockburn & Highsmith, 2001; Nerur, Mahapatra, & Mangalaraj, 2005), democratic (Siakas & Siakas, 2007), less formalized and non-hierarchical (Strode, Huff, & Tretiakov, 2009), and has an appropriate reward system (Derby, 2006). However, changing an existing organizational or department-level

* Corresponding author.
E-mail addresses: mgupta@fiu.edu (M. Gupta), jfgeorge@iastate.edu (J.F. George), xiaw@fiu.edu (W. Xia).

https://doi.org/10.1016/j.ijinfomgt.2018.09.006
Received 26 June 2018; Received in revised form 16 September 2018; Accepted 16 September 2018
Available online 25 September 2018

Table 1
Summary of literature related to categorization and descriptions of the Agile practices.

Category (Reference)	ASD Practice	Description
Technical (Maruping et al., 2009)	Unit testing	Programmers continually write unit tests, which must run flawlessly for development to continue (Beck, 2006).
Technical (Maruping et al., 2009)	Continuous Integration	Integrate and build the system many times a day, every time a task is completed (Beck, 2006).
Technical (Maruping et al., 2009)	Refactoring	The design of the system is evolved through transformations of the existing design that keep all the tests running (Beck, 2006).
Technical (Maruping et al., 2009)	Collective Ownership	Every programmer improves any code anywhere in the system at any time if they see the opportunity (Beck, 2006).
Technical (Maruping et al., 2009)	Coding Standards	Programmers write all code in accordance with rules emphasizing communication through the code.
Social (Hummel et al., 2015; Robinson & Sharp, 2005b)	Pair-programming	Two programmers form a team and constantly interact with each other to write all production code at one machine (Beck, 2006).
Social (Hummel et al., 2015; So & Scholl, 2009)	Customer/ Product Owner Role	The person responsible for articulating the product vision. This person actively works with other members to clear any issue pertaining to product features/requirements during systems development. He/she is the voice of the customer/end-user (Sutherland & Schwaber, 2007).
Social (Hummel et al., 2015; So & Scholl, 2009)	Daily Standup	A short meeting (time-boxed to 15 min) that takes place every day at the same time in which individuals give a daily status of their assigned tasks (Sutherland & Schwaber, 2007).
Social (Hummel et al., 2015; So & Scholl, 2009)	Retrospectives	A meeting that is used to discuss questions such as "what went wrong" and "what went well" in the past development cycle. It helps identify "what could be improved" in future development cycles (Sutherland & Schwaber, 2007).

culture is a challenging task (Iversen & Ngwenyama, 2006). Moreover, if at all the culture changes, it will be dependent on the existing organizational or department-level culture (Barney, 1986). While it is important to understand what the elements of an ideal agile culture are, even more important is to understand how existing IT department culture will affect the usage of ASD practices (Iivari & Iivari, 2011; Tolfo, Wazlawick, Ferreira, & Forcellini, 2011). Therefore, the primary research question that we examine in this study is: *How does IT department culture affect the use of social and technical agile practices?*

Our focus in the remainder of the paper is to explore the literature that informs our theory development and testing, the methods we use to collect and examine our data, and to examine and discuss the results of our investigation. We conclude the paper with implications of our research for theory and practice, limitations of the study, and directions for future research.

2. Theoritical background

2.1. ASD practices

Agile values and principles, as described by the Agile Manifesto, prescribe the overarching philosophies and guidelines underlying ASD. ASD methods are composed of various ASD practices. For example, XP is composed of such ASD practices as refactoring, collective ownership, and continuous integration (Beck, 1999). Since different agile methods were proposed out of particular circumstances with different and sometimes contradictory practices (Tripp & Armstrong, 2014), most organizations adopt a mix of ASD practices derived from two or more ASD methodologies (Conboy & Fitzgerald, 2010; Stavru, 2014; VersionOne, 2015). In this paper, we focus on ASD practices, which refer to specific techniques or actions taken by IT department in their routines of software development (Hummel et al., 2015).

Over the years, several scholars have suggested classifying ASD practices into technical and social types. For instance, Chow and Cao (2008) label software engineering-oriented practices, such as well-defined coding standards, up front and rigorous refactoring activities, as technical and practices fostering customer involvement and team motivation as social. Some suggest that, while several ASD practices are used during software development, some are focused on writing source code and test scenarios, whereas others facilitate communication and knowledge sharing among individuals (McHugh et al., 2011; Treude & Storey, 2009). Robinson and Sharp (2005b) assert that some ASD practices tend to be more socially-oriented than technically-oriented. Similarly, Mnkandla and Dwolatzky (2007) contend that ASD practices that relate to software coding are technical, while those relate to people issues are social. Jyothi and Rao (2011) suggest classifying the eleven agile principles of the agile manifesto in terms of technical and social dimensions. Vavpotic and Bajec (2009) suggest that, in order to understand the benefits of using a software development methodology, it is important to view software development through the lens of technically and socially-oriented ASD practices because focusing on either one will result in an incomplete evaluation of software development methodologies. While few studies have also used the label of project management practices in lieu social practices, (Tripp & Armstrong, 2014), the label project management, in general, refers to hierarchical planning, monitoring, and control of activities and resources in the academic literature (Hallinger & Snidvongs, 2008)

To summarize, there is a high degree of agreement among the scholars pertaining to the categorization of ASD practices into social and technical types (Corvera Charaf, Rosenkranz, & Holten, 2013; Diegmann & Rosenkranz, 2016; Ozcan-Top & Demirörs, 2013). Using this stream of literature as a theoretical basis, Hummel et al. (2015), "recognizing the importance of social interactions, social behavior, and communication in the ISD [information systems development] process," proposed the construct of social agile practices to describe a subset of ASD practices that foster interactions, collaboration, and direct communication among individuals (p. 280). By comparison, the subset of ASD practices that emphasizes the software engineering-oriented aspects of software development is defined by the technical agile practices construct.

We utilize the same definitions of the social and technical agile practices constructs in this study at the IT department-level, and specifically focus on six XP practices (unit testing, continuous integration, refactoring, collective ownership, coding standards, and pair-programming) and three widely-adopted Scrum practices (product owner, daily standups, and retrospectives). We choose these nine practices as they are not only widely used in industry, but also in the academic literature (Maruping, Venkatesh, & Agarwal, 2009; McHugh et al., 2011; So & Scholl, 2009). Following a number of studies discussed previously, we classified continuous integration, collective ownership, unit testing, refactoring, and coding standards as technical agile practices (Maruping et al., 2009), while daily-stand-ups, retrospectives, and

pair-programing were classified as social agile practices (Hummel et al., 2015; So & Scholl, 2009). Table 1 presents a summary of literature related to categorization and descriptions of the ASD practices.

2.2. IT department culture

Organizational culture is extremely important for a successful implementation of any new process (Desouza & Evaristo, 2006). While a number of organizational cultural frameworks have been proposed in the management literature, in this study, we adopt Quinn and Rohrbaugh (1983) competing-values model (CVM). CVM is one of the most established and well-cited cultural frameworks that has been developed from over 25 years of scholarly research (Cameron, Quinn, DeGraff, & Thakor, 2014; Vick, Nagano, & Popadiuk, 2015). We specifically chose CVM in this study for two reasons. First, this framework has been actively employed in the past studies on software development (e.g., Iivari & Huisman, 2007; Iivari & Iivari, 2011; Strode et al., 2009). Second, the CVM considers that a department within an organization may develop a different sub culture, and as a consequence, this framework can be applied at the department-level such as the IT department.

CVM is centered on the two pairs of opposing dimensions: (1) *internal focus versus external focus*; and (2) *flexibility versus stability* (Quinn & Rohrbaugh, 1983). While one dimension describes the differences between the values pertaining to the welfare of its employees (i.e., internal focus) versus the welfare of the organization itself (i.e., external focus), the other differentiates organizations based on the degree of structure (i.e., flexibility versus stability). As shown in Table 2, the intersections between these two opposing dimensions form four quadrants – hierarchical, rational, group, and developmental - each representing a different type of subculture.

Hierarchical culture lies at the intersection of stable organizational structure and internal focus. The origin of this form of culture dates back to the early 1900s when the main challenge faced by organizations was to efficiently develop products and services for a relatively predictable and stable market demand. The underpinnings of a hierarchical culture have primarily been based on the Weber (1947) work on bureaucracy - processes, measurement, and control. Hierarchical culture manifests standardized procedures to increase "the regularity and consistency of outcomes" (Cameron et al., 2014, p. 33). Rules are enforced and leaders establish uniformity across the internal affairs of the organization. Moreover, promotion of employees is dependent on the extent to which they understand the organization's rules and policies.

Rational culture lies at the intersection of external focus and stability. This form of culture is primarily based on the work of Ouchi and Johnson (1978) and Williamson (1981) theory of transaction cost economics. Rational culture emerged in the late 1960s as markets became more competitive. Consequently, the orientation of organizations changed from being methodical to being competitive by lowering transaction costs. Unlike hierarchical culture, rational culture manifests external focus such that plans and strategies are devised by carefully scrutinizing the external market in which the organization operates; however, control is exercised throughout the organization to ensure attainment of well-defined objectives.

Group culture lies at the intersection of internal focus and flexibility. This form of culture became evident after a number of management scholars in 1970s identified fundamental differences between the US and Japanese organizations (Cameron & Quinn, 2011). While organizations in the US functioned as economic entities, Japanese firms resembled clan-like groups of family members. The primary focus of a group culture is on the wellbeing of its employees rather than on the measures of financial performance (Quinn & Rohrbaugh, 1983).

Developmental culture is characterized by an external focus and a flexible organizational structure. This form of culture originated as organizations faced heightened uncertainty in the 21 st century (Cameron & Quinn, 2011). Developmental culture enables organizations to operate in hyper-turbulent market conditions (Cameron & Quinn, 2011). Developmental culture manifests risk-taking and thoughtful experimentation, focuses on being innovative, and views failures as learning opportunities rather than as embarrassments. It is characterized by forward-looking and continuous explorations of new growth opportunities. Customers' feedback is actively sought out to improve existing products and to elicit new ideas.

3. Research hypotheses

3.1. IT department culture and ASD practices

Culture plays a critical role in "determining patterns of IT development, adoption, use, and outcomes" (Leidner & Kayworth, 2006, p. 381). Cockburn and Highsmith (2001) assert that the extent to which ASD practices are adopted in an IT department depends on its existing culture. Fruhling and Tarrell (2008), based on qualitative and quantitative data collected from individuals involved in ASD, found IT department culture as the most significant barrier to successful ASD implementation. Cao et al. (2009) conducted a case study of four IT projects across three different organizations that had adopted ASD using XP. Their analysis revealed that the prevailing IT department culture can exhibit (or inhibit) the use of ASD practices. For instance, in one of the cases, forcing developers to engage in pair-programming resulted in conflicts.

A number of other studies (e.g., Iivari & Iivari, 2011; Nerur et al., 2005; Robinson & Sharp, 2005a; Siakas & Siakas, 2007; Strode et al., 2009; Tolfo & Wazlawick, 2008) have also proposed the influence of culture on the usage of ASD practices. While the existing literature yields interesting insights into the relationship between IT department culture and ASD practices usage, the majority of these studies have been either conceptual or exploratory in nature. Additional research is needed to develop insights on how different forms of culture affect the adoption of different types of ASD practices, such as social and technical, at the IT department-level.

3.1.1. Hierarchical culture and ASD practices

The focus of hierarchical culture is to ensure smooth (and predictable) running of day-to-day organizational operations. Anything that could potentially cause a failure or result in an uncertainty is considered a threat and is immediately eliminated. Projects that are not productive are often divested and employee layoffs are common. Leadership style is top down such that employees are required to follow a predetermined plan approved by their superiors. Hierarchical culture

Table 2
Differences across four competing cultural forms.

Organizational Culture	Emergence	Focus	Structure	Motivation Factors	Leadership Style	Measure of Success
Hierarchical	Early 1900s	Internal	Stable/ Controlled	Rules, Regulations	Authoritarian	Efficiency
Rational	Late 1960s	External	Stable/ Controlled	Meeting deadlines	Directive, Goal-Oriented	Financial Performance
Group	1970s	Internal	Flexible	Cohesiveness, We-ness	Parent Figures, Mentors	Human Relations
Developmental	Early 2000s	External	Flexible	Not afraid of failures	Risk Takers, Visionaries	Growth

constricts the flow of information. Feldman and March (1981), in their seminal work, called information a tool for obtaining power. Employees, especially those in leadership roles, tend to withhold information since having more information than others is considered a symbol of power. Hierarchical organizations are conducive to stability.

Social agile practices encourage member participation and employee interactions, and advocate free flow of information at all levels in the software development process. Practices, such as daily standups and retrospective meetings, enable employees at different levels and in different roles to voice their concerns and suggest improvements in the development cycle (Harvie & Agah, 2016). Since a hierarchical culture follows an authoritative top-down leadership style where employees are expected to follow the instructions of their superiors, it is likely that employees may be less forthcoming. The authoritative top-down leadership style where employees are expected to follow the instructions of their superiors leads to vertical communication and inhibits lateral communication, which is endorsed by social agile practices. Consequently, hierarchical form of IT department culture is less likely to encourage the use of social agile practices in its software development.

H1. Hierarchical culture will have a negative impact on social agile practices usage.

A major emphasis of a hierarchical organization is to efficiently deliver products (or services) to its customers by using a set of standardized uniform processes. Hierarchical culture endorses following a plan and is not open to changes once initial requirements are decided. Part of the reason for this is that hierarchical culture tends to be highly internally focused such that changes in the external environment are considered less important than maintaining efficiency in daily operations. Furthermore, hierarchical culture is detail-oriented where enormous amount of time and resources are spent on documenting even the smallest details. By comparison, technical agile practices emphasize on writing code that works rather than extensively documenting all possible details (e.g., project requirements, system design, user acceptance tests, etc.) and events in the software development process. It does not mean that no documentation takes place when technical agile practices are followed; instead, minimal or "just enough" documentation is encouraged (Law & Charron, 2005). Since the tenets of hierarchical culture are less likely to foster the use of technical agile practices, we hypothesize:

H2. Hierarchical culture will have a negative impact on technical agile practices usage.

3.1.2. Rational culture and ASD practices

Software development is a complex process that requires involvement of several individuals and effective communication among them (Lin, Chen, Hsu, & Fu, 2015; Park & Lee, 2014; Recker, Holten, Hummel, & Rosenkranz, 2017). One of the major requirements for implementing social agile practices is a presence of culture that allows employees in the IT department to indulge in discussions via face-to-face interactions. For instance, social agile practices such as access to customer, daily standups, retrospective meetings, and pair-programming provide an effective face-to-face platform for individuals to converse and discuss project activities. Rational culture, like hierarchical culture, emphasizes control and stability in the departmental structure. The assertive leadership style leaves little room for employees to freely share their opinions or indulge in discussions, especially, when they do not agree with the perspective of their superiors. Rational culture lacks equal member participation. What matters the most for rational organizations is to be profitable in the near term. Thus, in rationally-oriented IT departments, social agile practices-fostered face-to-face interactions may be perceived as loss of important time of its employees and disturbance and distraction for other individuals (Mishra, Mishra, & Ostrovska, 2012). Therefore, we hypothesize:

H3. Rational culture will have a negative impact on social agile practices usage.

In comparison to social agile practices, rational culture will likely be more conducive to the use of technical agile practices in software development efforts. Due to its external focus, rational culture encourages following the activities of competitors and strive to outperform them. While rational culture is less likely to promote exploring new growth opportunities outside of the niche, it actively stimulates ways to increase their market share. A rational culture treats markets as hostile, and the only way to succeed is to defeat the opposition. Using technical agile practices in the IT department allow organizations to remain competitive by developing and releasing software in Internet time (i.e., fast to the market) (Iansiti & MacCormack, 1996). Technical agile practices such as refactoring improves the quality of software by continuously reviewing the existing design of the system while keeping its core functionality intact. Thus, any future or unplanned changes in software requirements are comparatively easier to implement than when technical agile practices are not followed. This is particularly important for market-oriented organizations, which often need flexibility in their software development efforts due to the prevailing market uncertainties and the fear of losing market share.

H4. Rational culture will have a positive impact on technical agile practices usage.

3.1.3. Group culture and ASD practices

Group-oriented culture values employee development and strives to provide its employees with a healthy work-life balance. Employees have a sense of collectiveness and tend to be loyal to their organizations. Leaders are considered parent figures and mentors rather than administrators. Communication is not controlled and information is freely dispersed across the whole organization. Employee participation is encouraged and employees have easy access to their immediate superiors and other leaders.

Strong interpersonal skills and an environment that fosters social interactions among employees are important characteristics of a group culture. The work place environment tends to be friendly and conflicts are avoided by achieving consensus. It is due to this reason that group culture is also termed as a people-oriented culture. In this regard, the values endorsed by group culture are in line with the culture that is needed to promote the use of social agile practices. Specifically, a family-like environment in the IT department will enable individuals to participate in social agile practices, such as talking to the customer or participating in retrospective meetings, without being fearful of their superiors or peers. Unlike hierarchical and rational cultures, group culture does not have communication barriers, absence of which is likely to encourage social agile practices-fostered interactions between technical and management staff.

H5. Group culture will have a positive impact on social agile practices usage.

A group culture-oriented workplace tends to be more concerned about developing long-lasting human relationships within and outside of its boundary. Although the group culture endorses investing in employee development, the roles and responsibilities of employees are well-defined and fit between task requirements and employees' skills is ensured (Cameron et al., 2014). However, technical agile practices require individuals to have good generalized skills, in addition to specialized skills, which allow ASD teams to respond to changing external requirements without delay (Nerur & Balijepally, 2007; vom Brocke, Zelt, & Schmiedel, 2016). It can be expected of developers to test and there may be times when individuals in testing roles are expected to write code. Nerur and Balijepally (2007) liken this to Ashby's law of requisite variety where "a system's internal variety should at least match the variety and complexity of the environment with which it is

Table 3
Sample direct quotes.

Social involves more interactions face-face and trying to move the needle, while technical are tool based, systemic engineering practices.
Technical - software-based; Social - people-based
Things that "touch" the code are technical, and things that "touch" other people are social.
social requires communication; technical not that much
The stuff I marked as "Technical" is stuff developers do or need to do to produce verifiably "working software"–the heart of what we need and again, the primary measure of progress. The rest I marked as social…activities (practices) that help everyone else know the relative state of value delivery….code is king and that stuff I marked "technical".
Social emphasis to me is about anything that fosters or supports collaboration to create better code whereas technical is internally focused on helping the individual developer to write better code
Social practices exist to move information around to benefit from the network of minds.
I categorized social practices the ones where interaction between people is needed or higher rated than the pure technical skills of an individual.
Social are interaction enabling and some are pure technical
Practices involving communication are social, while ones that involves technical components are technical
All practices that involve more than one person are social in my opinion.
Each time some interactions is mandatory to help the team or one or two team's members to better understand what is wrong or what is good, to understand what is awaited is for me à social type. All others are technical. Technical practices are only ways of writing/developing working software
I classified as social all practices which involve more than one people as direct actor or involved/affected party.
The Social dimension focusses on the conversation, the technical about the craftsmanship
Any meeting related was categorized social while any coding related technical
Anything that requires two or more developers to directly interact would be social. Anything else is technical.
Social: people and interactions, while Technical - refer to practices that promote coding quality, standards and best practices, e.g. "how to best write the code"…
I classified anything that could be done alone as "Technical" and those that were typically with groups as Social.

confronted" (p. 83). Group culture emphasizes specialized employee skills, and thus it is highly likely that interchangeability of roles or jobs may not be encouraged in the software development process. Moreover, group culture-oriented IT departments may face criticism if their employees are asked to learn technical agile practices, especially if using technical agile practices is not a part of their defined roles and responsibilities. Based on this:

H6. Group culture will have a negative impact on technical agile practices usage.

3.1.4. Developmental culture and ASD practices

Developmental culture has no centralized leadership, and thus can be considered a complete opposite of hierarchical culture. While leaders in hierarchical culture consider information as a tool for obtaining power, developmental leaders ensure that information is passed on to all employees regardless of their job titles. Communication is unrestricted and information is not controlled.

Like group culture, developmental culture encourages employee participation. However, while family-like environment in a group culture fosters employee participation, the forward-looking nature of the development culture inspires its employees to actively engage in conversations that could potentially stimulate new thoughts. Developmental culture, unlike rational culture, does not consider social interactions as a waste of employee time; instead, consistent with social agile practices, social interactions are considered enablers for encouraging employee participation, dispersing information, and facilitating communication within and between groups (or departments). In sum, developmental culture is likely to promote the use of social agile practices that allow individuals to clarify system requirements, share their expertise, and update each other about the overall systems development progress.

H7. Developmental culture will have a positive impact on social agile practices usage.

Developmental culture embraces that the external environment is uncertain and ambiguous and they need to frequently adapt to changing market conditions. Unlike hierarchical culture, which focuses on maintaining uniformity, developmental culture focuses on double loop learning (i.e., learning to learn). Developmental culture encourages experimentation and is not afraid of failures. As discussed previously, developmental culture promotes evaluating and identifying future trends in and out of their principal industry.

Following technical agile practices will enable IT departments to quickly create and release novel software features (or products) that in turn will allow the organizations to stay ahead of the competition in their principal industry and at the same time allow them to explore opportunities outside of their principal industry. Moreover, employees under developmental culture tend to have both specialized and generalized skills and their roles and responsibilities are defined based on the current circumstances. It is due to this reason that developmental culture is often termed as an organized anarchy. Given the flexibility that technical agile practices provide in the software development process, developmental culture is likely to encourage the use of technical agile practices.

H8. Developmental culture will have a positive impact on technical agile practices usage.

4. Research methodology

We further validate the categorization of nine ASD practices that we focus in this study into social and technical types by conducting an online Q-sorting exercise, consisting of 47 agile coaches/mentors, The participants represented a variety of industries, such as computers and finance and other industries, such as manufacturing, healthcare, and utilities, and had an average of 10 years of experience in ASD. The participants were first provided with a description of social and technical agile practices. They were then asked to assign each of the nine ASD practices into either social or technical types. The order of the nine ASD practices was randomized so that no two participants viewed the nine ASD practices in the same order. After assigning each of the nine ASD practices into one of the two categories, the participants were asked to provide feedback on their views of the two categories, and if they did not agree, to propose alternative categories. They also provided rationales based on which they assigned the ASD practices into the two categories. Some direct quotes from the participants are illustrated in Table 3.

For eight of the nine practices, there was a clear consensus among participants and it matched with our categorization of ASD practices into either social or technical categories – unit testing (44 technical, 3 social), collective ownership (39 technical, 8 social), refactoring (43 technical, 4 social), coding standards (38 technical, 9 social), and continuous integration (40 technical, 7 social), daily stand-ups (2 technical, 45 social), retrospectives (1 technical, 46 social), and product owner role (0 technical, 47 social). However, for pair-programming, about half of the experts placed it under the technical category, while the remaining half classified it as social. For instance, according to one respondent, "since it [pair-programming] was typically two developers working on code, I marked it as technical." In contrast, others opined that the main purpose of pair-programming is to enable social interactions between developers with complementary skills and perspectives. Given the lack of consensus of the Q-sorting results for pair-programming, we relied on extant literature in classifying pair-programming as social or technical.

We found that the majority of recent and past ASD studies have conceptualized pair-programming as an enabler of social interactions

between software developers (Berente & Lyytinen, 2007; Hummel et al., 2015; Salge & Berente, 2014; Williams & Kessler, 2002). This is consistent with Robinson and Sharp (2005b) who indicated that "perhaps the most striking thing about the observed pairing was that it was not so much about code as about conversation" (p. 102). Similarly, Storey, Zagalsky, Singer, and German (2015) talk about the rise of the social programmer, who does not only code, but also "actively engage with, learn from and co-create with other developers" (p. 1). In the same vein, Pedrycz, Russo, and Succi (2011), and Kent Beck (1999), a distinguished ASD practitioner and the creator of XP, indicate that the primary purpose of pair-programming is to stimulate social interactions because it forces developers to talk with each other. Following this stream of literature, we categorized pair-programming as a social agile practice in this study.

4.1. Construct measures

We used the well-established measures from the extant literature. The four CVM cultural constructs were measured using the scales proposed by Iivari and Huisman (2007). IT department's usage of technical agile practices were measured with scales proposed by Maruping et al. (2009), while social agile practices construct was measured using the scales suggested by So and Scholl (2009). All questions were asked using a 7-point Likert scale (1 = Strongly Disagree and 7 = Strongly Agree). Questions capturing participant's industry and the IT department's size were also included as controls as suggested by prior studies on software development (Iivari & Huisman, 2007).

Before administering the survey questions to a large group of participants, we first conducted a pilot test to validate the measures of nine ASD practices and four cultural types measured at the IT department-level. According to MacKenzie, Podsakoff, and Podsakoff (2011), it is critical for survey studies to first validate the measures adopted from other studies before collecting data for hypotheses testing. The moderator of a local agile user group sent the link to the pilot survey to agile project managers in the area. In total, 54 completed responses were received with respondents having an average of 4.2 years of agile experience. The Cronbach's α for most constructs were excellent (> .0.70), while it was satisfactory for rational culture (α = .60) and hierarchical (α = .64) culture constructs (see Table 4). Thus, all items from the pilot study were retained for the main study.

4.2. Study sample

To test our hypotheses, a survey was created in Qualtrics and the link to the survey was sent to a sample of 950 IT managers, who were located in the United States. The list was drawn from an online LinkedIn community of over 12,000 agile practitioners at the time of the study. We chose IT managers as participants for this study because past research suggests that they tend to be knowledgeable about the extent to which ASD practices are used in the IT department (Iivari & Huisman, 2007). In total, 179 usable responses (18.5% response rate) were received. Table 5 presents the characteristics of our main study sample.

We compared the responses of early respondents with late respondents on all constructs to test for nonresponse bias. The analysis revealed no statistically significant differences between the early and the late responses: group culture (t = 0.855, ns), developmental culture (t = 0.604, ns), rational culture (t = 0.982, ns), hierarchical culture (t = 0.416, ns), refactoring (t = 0.234, ns), continuous integration (t = 0.55, ns), unit testing (t = 0.055, ns), pair-programming (t = 1.489, ns), retrospectives (t = 0.183, ns), stand-ups (t = 0.186, ns), and access to the product owner (t = 0.884, ns).

We performed Harman's single-factor test on all constructs to ensure that common method bias due to a single response was not a significant issue (Podsakoff & Organ, 1986). All constructs were entered into an exploratory factor analysis using principal component analysis. The number of factors was then determined by examining the original (unrotated) factor solution. A single factor should account for the majority of covariance in the latent constructs if there exists a significant common method bias (Podsakoff & Organ, 1986). The analysis did not reveal a single dominating general factor.

5. Results

5.1. Measurement validation

Both technical and social agile practices constructs were modelled as second-order constructs. The technical agile practices construct consisted of five first-order constructs - continuous integration, collective ownership, coding standards, unit testing, and refactoring, while the social agile practices construct consisted of four first-order constructs - pair-programming, customer access, retrospectives, and stand-ups. We followed a two-stage method to validate the measures of technical and social agile practices constructs (Hair, Hult, Ringle, & Sarstedt, 2013). In the first stage, the first-order measurement model was assessed, and the psychometric properties of the first-order constructs were evaluated. Then, using the latent variable scores of the five technical agile practices and four social agile practices constructs, the second-order structural model was estimated.

We used PLS-SEM to evaluate the first-order and second-order models. For the first-order measurement model, we first examined indicators' outer loadings. All indicators, with the exception of HC3 and CS3, had significant outer loadings. As a result, HC3 and CS3 were dropped, while the remaining items were retained (Hair et al., 2013). Descriptive statistics of the measures are shown in Table 6, while reliabilities are displayed in Table 7.

We then assessed the convergent and discriminant validity of all the first-order constructs (see Table 7). An average variance extracted (AVE) value above 0.50 establishes the convergent validity of a latent construct (Hair et al., 2013), while the square root of the AVEs of each construct greater than its correlation with any other construct provides evidence for the discriminant validity (Fornell & Larcker, 1981). The values reported in Table 7 demonstrate excellent composite reliabilities (CR) and Cronbach's α, with the exception of group culture, rational culture, and continuous integration constructs for which the α values were satisfactory according to Hair and Hult (2016). All AVE values were above 0.50, suggesting convergent validity. The diagonal elements, which were the square roots of the AVEs were greater than the construct's correlations with any other constructs, establishing the discriminant validity of the measures. Discriminant validity was further assessed by examining items' cross loadings (see Table 8), which were all found to be smaller than their factor (of interest) loadings (Hair et al., 2013). In sum, the first-order measurement model demonstrated good psychometric properties.

We then built the higher-order models for technical and social agile practices, using the standardized latent variable scores obtained from the first-order measurement model for each of the five technical agile practices and the four social agile practices constructs, respectively. As shown in Fig. 1, technical and social agile practices constructs were modelled as reflective constructs, which is consistent with the past studies, such as Maruping et al. (2009) and Hummel et al. (2015), who have modelled second-order constructs with ASD practices as reflective indicators.

Moreover, an important step in identifying whether a construct is reflective or formative is to assess whether its indicators share a common theme or they collectively form the construct (Petter, Straub, & Rai, 2007). The answer to this issue also depends upon how a researcher has defined the construct because a construct is not inherently reflective or formative (MacKenzie et al., 2011). Consistent with prior literature (Hummel, 2013; Hummel et al., 2015), our definition of the technical agile practices construct reflect a common "technical" theme among the practices of continuous integration, collective ownership, unit testing, refactoring, and coding standards. By comparison, our

Table 4
Descriptive statistics of pilot data (N = 54).

Construct (Reference)		Item	Mean	SD	α
Continuous Integration (CI)	CI1	We integrate newly coded units of software with existing code.	5.98	1.09	.73
Maruping et al. (2009)	CI2	We combine new code with existing code on a continual basis.	6.11	1.02	
	CI3	We do not take time to combine various units of code as they are developed.	5.22	1.63	
Collective Ownership (CO)	CO1	Anyone can change existing code at any time.	4.20	2.03	.76
Maruping et al. (2009)	CO2	If anyone wants to change a piece of code, they need the permission of the individual(s) that coded it.	5.54	1.41	
	CO3	We are comfortable changing any part of the existing code at any time.	4.11	1.75	
Coding Standards (CS)	CS1	We have a set of agreed upon coding standards in this team.	5.30	1.50	.75
Maruping et al. (2009)	CS2	We have a shared understanding of how code is to be written.	5.00	1.49	
	CS3	We all use our own standards for coding.	4.48	1.71	
Unit Testing (UT)	UT1	We run unit tests on newly coded modules until they run flawlessly.	5.26	1.58	.80
Maruping et al. (2009)	UT2	We actively engage in unit testing.	5.74	1.38	
	UT3	To what extent are unit tests run by the members in this department?	5.46	1.60	
Refactoring (REF)	REF1	Where necessary, we try to simplify existing code without changing its functionality.	5.04	1.43	.89
Maruping et al. (2009)	REF2	We periodically identify and eliminate redundancies in the software code	4.78	1.62	
	REF3	We periodically simplify existing code	4.57	1.64	
Pair Programming (PP)	PP1	How often is pair programming used in this department?	4.17	2.04	.92
Maruping et al. (2009)	PP2	We do our software development using pairs of developers	3.54	1.96	
	PP3	To what extent is programming carried out by pairs of developers in this department	3.76	2.02	
Stand-ups (SU)	SU1	Stand up meetings are extremely short (max. 15 min).	5.41	1.54	.72
So and Scholl (2009)	SU2	Stand up meetings are to the point, focusing only on what has been done and needed to be done on that day.	5.09	1.47	
	SU3	All relevant technical issues or organizational impediments come up in the stand-up meetings.	4.87	1.47	
	SU4	Stand up meetings provide the quickest way to notify other members about problems.	5.33	1.51	
	SU5	When people report problems in the stand-up meetings, other members offer to help instantly.	5.17	1.26	
Retrospectives (RET)	RET1	We actively participate in gathering lessons learned in the retrospectives.	5.24	1.66	.95
So and Scholl (2009)	RET2	The retrospectives help us become aware of what we did well in the past iteration/s.	5.30	1.67	
	RET3	The retrospectives help us become aware of what we should improve in the upcoming iteration/s.	5.31	1.70	
	RET4	In the retrospectives (or shortly afterwards), we systematically assign all important points or improvement to responsible individuals.	4.57	1.77	
	RET5	We follow up intensively on the progress of each improvement point elaborated in a retrospective.	4.15	1.57	
Customer Access (CA)	CA1	The customer is reachable.	5.04	1.64	.90
So and Scholl (2009)	CA2	The developers can contact the customer directly or through a customer contact person without any bureaucratic hurdles.	4.61	1.87	
	CA3	The developers have responses from the customer in a timely manner.	4.74	1.62	
	CA4	The feedback from the customer is clear and clarified about the requirements or open issues to the developers.	4.54	1.68	
Group Culture (GC)	GC1	The glue that holds the IT department I work in together is loyalty and tradition.	4.54	1.56	.81
Iivari & Huisman, 2007	GC2	The IT department I work in is a very personal place.	4.44	1.42	
	GC3	The IT department I work in emphasizes human resources.	4.43	1.60	
Developmental Culture (DC)	DC1	The IT department I work in is a very dynamic and entrepreneurial place.	4.54	1.56	.72
Iivari & Huisman, 2007	DC2	The glue that holds the IT department I work in together is commitment to innovation and development.	4.24	1.53	
	DC3	The IT department I work emphasizes acquiring new resources and meeting new challenges.	4.43	1.28	
Rational Culture (RC)	RC1	The glue that holds the IT department I work in together is the emphasis on tasks and goal accomplishment.	5.11	1.14	.60
Iivari & Huisman, 2007	RC2	The IT department I work in is a very production-oriented place.	4.33	1.36	
	RC3	The IT department I work in emphasizes competitive actions, outcomes and achievement.	4.63	1.28	
Hierarchical Culture (HC)	HC1	The IT department I work in is a very formal and structured place.	3.72	1.57	.64
Iivari & Huisman, 2007	HC2	The glue that holds the IT department I work in together is formal rules and policies.	4.46	1.42	
	HC3	The IT department I work in emphasizes permanence and stability.	4.41	1.30	

Table 5
Main study sample characteristics.

Industries	
Computers/Software	37.8%
Finance (Banking/Real Estate)	21.8%
Others (Manufacturing, Healthcare, Utilities, Publishing, etc.)	27.6%
Total agile experience of the respondents	
Less than 5 years	43.1%
5 to 10 years	40.0%
More than 10 years	16.9%
Number of employees in the organization	
Fewer than 1000	44.4%
1,000–10,000	26.3%
More than 10,000	29.3%
Number of employees in the IT department	
Fewer than 100	61.3%
100 to 500	21.8%
More than 500	16.9%

definition of the social agile practices construct suggests a common underlying "social" theme among the practices of daily-stand-ups, pair-programming, retrospectives, and customer access. Likewise, Hummel et al. (2015) justified the use of the term "social" in the social agile practices construct to acknowledge that all the practices reflected by this construct relate to [social] interaction.

In addition to this, Cronbach's αs and CR values for both second-order social and technical agile practices construct were excellent (see Table 9). According to Petter et al. (2007), multicollinearity, which is highly desirable for the reflective constructs are "illustrated by high Cronbach's α and composite reliability scores" (p. 641). In sum, both our conceptual definitions of the social and technical agile practices construct and their high reliability scores provide support that it is appropriate to model these constructs as reflective (Jarvis, MacKenzie, & Podsakoff, 2003; Petter et al., 2007).

We then estimated the second-order measurement model. The indicators' loadings, including those of the second-order constructs, were above 0.6; however, the continuous integration (CI) indicator of the technical agile practices construct had an outer loading of .56. Following Wetzels, Odekerken-Schröder, and Van Oppen, (2009), we assessed the validity and reliability of the higher-order model by checking the composite reliabilities, Cronbach αs, and AVE values of all the constructs in the hierarchical model. All composite reliabilities were excellent and Cronbach αs were satisfactory. However, while all AVE values were above the recommended value of 0.5, the AVE for the

Table 6
Descriptive statistics of the measures for the main study (N = 179).

Construct		Item	Mean	SD
Continuous Integration (CI)	CI1	We integrate newly coded units of software with existing code.	6.09	.96
	CI2	We combine new code with existing code on a continual basis.	6.09	1.06
	CI3	We do not take time to combine various units of code as they are developed.	5.03	1.72
Collective Ownership (CO)	CO1	Anyone can change existing code at any time.	4.59	2.06
	CO2	If anyone wants to change a piece of code, they need the permission of the individual(s) that coded it.	5.59	1.55
	CO3	We are comfortable changing any part of the existing code at any time.	4.49	1.71
Coding Standards (CS)	CS1	We have a set of agreed upon coding standards in this team.	5.36	1.37
	CS2	We have a shared understanding of how code is to be written.	5.41	1.32
	CS3	We all use our own standards for coding.	3.48	1.78
Unit Testing (UT)	UT1	We run unit tests on newly coded modules until they run flawlessly.	5.40	1.42
	UT2	We actively engage in unit testing.	5.74	1.30
	UT3	To what extent are unit tests run by the members in this department?	5.51	1.43
Refactoring (REF)	REF1	Where necessary, we try to simplify existing code without changing its functionality.	5.70	1.15
	REF2	We periodically identify and eliminate redundancies in the software code	5.45	1.36
	REF3	We periodically simplify existing code	5.16	1.44
Pair Programming (PP)	PP1	How often is pair programming used in this department?	4.25	1.93
	PP2	We do our software development using pairs of developers	3.68	1.72
	PP3	To what extent is programming carried out by pairs of developers in this department	3.77	1.754
Stand-ups (SU)	SU1	Stand up meetings are extremely short (max. 15 min).	5.61	1.55
	SU2	Stand up meetings are to the point, focusing only on what has been done and needed to be done on that day.	5.39	1.42
	SU3	All relevant technical issues or organizational impediments come up in the stand-up meetings.	4.91	1.46
	SU4	Stand up meetings provide the quickest way to notify other members about problems.	4.97	1.73
	SU5	When people report problems in the stand-up meetings, other members offer to help instantly.	5.22	1.34
Retrospectives (RET)	RET1	We actively participate in gathering lessons learned in the retrospectives.	5.55	1.51
	RET2	The retrospectives help us become aware of what we did well in the past iteration/s.	5.75	1.35
	RET3	The retrospectives help us become aware of what we should improve in the upcoming iteration/s.	5.74	1.36
	RET4	In the retrospectives (or shortly afterwards), we systematically assign all important points or improvement to responsible individuals.	4.76	1.58
	RET5	We follow up intensively on the progress of each improvement point elaborated in a retrospective.	4.56	1.51
Customer Access (CA)	CA1	The customer is reachable.	5.35	1.64
	CA2	The developers can contact the customer directly or through a customer contact person without any bureaucratic hurdles.	4.85	1.89
	CA3	The developers have responses from the customer in a timely manner.	4.89	1.64
	CA4	The feedback from the customer is clear and clarified about the requirements or open issues to the developers.	4.54	1.68
Group Culture (GC)	GC1	The glue that holds the IT department I work in together is loyalty and tradition.	4.54	1.56
	GC2	The IT department I work in is a very personal place.	4.44	1.42
	GC3	The IT department I work in emphasizes human resources.	4.43	1.60
Developmental Culture (DC)	DC1	The IT department I work in is a very dynamic and entrepreneurial place.	4.54	1.56
	DC2	The glue that holds the IT department I work in together is commitment to innovation and development.	4.24	1.53
	DC3	The IT department I work emphasizes acquiring new resources and meeting new challenges.	4.43	1.28
Rational Culture (RC)	RC1	The glue that holds the IT department I work in together is the emphasis on tasks and goal accomplishment.	5.11	1.14
	RC2	The IT department I work in is a very production-oriented place.	4.33	1.36
	RC3	The IT department I work in emphasizes competitive actions, outcomes and achievement.	4.63	1.28
Hierarchical Culture (HC)	HC1	The IT department I work in is a very formal and structured place.	3.72	1.57
	HC2	The glue that holds the IT department I work in together is formal rules and policies.	4.46	1.42
	HC3	The IT department I work in emphasizes permanence and stability.	4.41	1.30

technical agile practices construct was 0.48. Using the recommendations of Hair and Hult (2016), we dropped the CI indicator of the second-order technical agile practices construct, and evaluated the model again. The resultant model (see Table 9) satisfied the quality criteria for convergent and discriminant validities, including all the AVE values above .5, as recommended in the literature (Hair & Hult, 2016; Hair, Black, Babin, Anderson, & Tatham, 2006; Wetzels et al., 2009). Additionally, the square root of the AVE values for each

Table 7
Reliabilities and correlations.

	α	CR	AVE	CA	CI	CO	CS	DC	GC	HC	PP	RC	REF	RET	SU	UT
CA	0.88	0.92	0.74	0.86												
CI	0.79	0.91	0.83	0.16	0.91											
CO	0.69	0.83	0.63	0.15	0.31	0.79										
CS	0.83	0.92	0.85	0.34	0.28	0.18	0.92									
DC	0.72	0.83	0.63	0.37	0.11	0.26	0.41	0.79								
GC	0.66	0.82	0.60	0.27	0.09	0.21	0.28	0.66	0.77							
HC	0.73	0.88	0.78	−0.15	−0.18	−0.45	−0.11	−0.26	−0.23	0.88						
PP	0.94	0.96	0.90	0.23	0.25	0.27	0.48	0.32	0.35	−0.15	0.95					
RC	0.60	0.76	0.52	0.23	0.01	−0.06	0.36	0.46	0.31	0.23	0.20	0.72				
REF	0.86	0.92	0.79	0.35	0.31	0.39	0.45	0.43	0.36	−0.28	0.45	0.18	0.89			
RET	0.91	0.94	0.74	0.44	0.10	0.09	0.49	0.43	0.38	−0.09	0.36	0.31	0.31	0.86		
SU	0.76	0.84	0.51	0.34	0.08	0.21	0.36	0.32	0.35	−0.20	0.24	0.24	0.35	0.57	0.72	
UT	0.85	0.91	0.77	0.25	0.43	0.29	0.43	0.27	0.21	−0.19	0.43	0.09	0.46	0.37	0.46	0.88

The square roots of average variance extracted (AVE) appear on the diagonals.
Notes: CA = Customer Access, CI = Continuous Integration, CO = Collective Ownership, CS = Coding Standards, DC = Developmental; GC = Group; HC = Hierarchical; PP = Pair-Programming, RC = Rational; REF = Refactoring, RET = Retrospectives; SU = Stand-up, and UT = Unit Testing.

Table 8
Cross-loadings.

	DC	GC	HC	RC	CI	CO	CS	REF	UT	CA	PP	RET	SU
DC1	**0.82**	0.55	−0.28	0.29	0.12	0.28	0.24	0.33	0.17	0.29	0.20	0.30	0.25
DC2	**0.89**	0.55	−0.10	0.45	0.15	0.23	0.42	0.46	0.27	0.39	0.32	0.41	0.27
DC3	**0.62**	0.48	0.04	0.28	−0.07	−0.03	0.21	0.13	0.07	0.22	0.09	0.26	0.15
GC1	0.42	**0.69**	−0.27	0.05	0.20	0.23	0.14	0.31	0.17	0.17	0.23	0.16	0.25
GC2	0.45	**0.71**	0.10	0.38	−0.04	0.07	0.19	0.23	0.13	0.20	0.21	0.28	0.22
GC3	0.63	**0.90**	−0.14	0.32	0.06	0.14	0.28	0.29	0.14	0.34	0.28	0.41	0.30
HC1	−0.13	−0.10	**0.84**	0.27	−0.21	−0.27	0.07	−0.23	−0.07	−0.05	−0.13	0.02	−0.10
HC2	−0.16	−0.14	**0.93**	0.26	−0.22	−0.45	−0.13	−0.22	−0.12	−0.11	−0.13	−0.11	−0.26
RC1	0.29	0.18	0.23	**0.76**	−0.07	−0.04	0.29	0.25	0.09	0.29	0.14	0.21	0.26
RC2	0.30	0.15	0.17	**0.78**	0.04	−0.06	0.30	0.06	0.09	0.16	0.24	0.21	0.11
RC3	0.40	0.41	0.25	**0.65**	−0.02	−0.05	0.23	0.10	0.05	0.21	0.13	0.23	0.09
CI1	0.15	0.15	−0.15	0.07	**0.85**	0.27	0.23	0.29	0.44	0.15	0.24	0.17	0.11
CI2	0.13	0.10	−0.18	−0.01	**0.82**	0.33	0.23	0.30	0.30	0.08	0.26	0.09	0.12
CI3	−0.01	−0.04	−0.24	−0.14	**0.59**	0.12	0.11	0.19	0.21	0.23	−0.04	0.13	0.09
CO1	0.16	0.14	−0.40	−0.12	0.31	**0.89**	0.12	0.33	0.20	0.09	0.20	0.06	0.21
CO2	0.09	0.08	−0.30	−0.11	0.28	**0.63**	−0.07	0.13	0.06	0.05	0.05	−0.07	0.08
CO3	0.30	0.20	−0.30	0.05	0.20	**0.83**	0.24	0.47	0.33	0.21	0.25	0.23	0.33
CS1	0.31	0.19	0.04	0.35	0.17	0.04	**0.90**	0.36	0.41	0.30	0.39	0.41	0.29
CS2	0.39	0.30	−0.12	0.34	0.28	0.21	**0.93**	0.48	0.41	0.38	0.46	0.50	0.34
REF1	0.42	0.32	−0.27	0.14	0.35	0.34	0.43	**0.88**	0.48	0.31	0.46	0.40	0.35
REF2	0.37	0.30	−0.20	0.23	0.30	0.40	0.45	**0.90**	0.42	0.32	0.42	0.26	0.37
REF3	0.35	0.32	−0.19	0.17	0.27	0.35	0.34	**0.87**	0.38	0.33	0.38	0.21	0.30
UT1	0.20	0.14	−0.08	0.07	0.34	0.24	0.40	0.43	**0.87**	0.28	0.45	0.32	0.40
UT2	0.20	0.18	−0.12	0.08	0.42	0.21	0.42	0.46	**0.92**	0.38	0.36	0.35	0.45
UT3	0.25	0.18	−0.09	0.14	0.36	0.25	0.37	0.42	**0.87**	0.27	0.35	0.34	0.44
CA1	0.29	0.26	0.01	0.21	0.18	0.00	0.31	0.27	0.25	**0.82**	0.21	0.40	0.28
CA2	0.37	0.30	−0.13	0.24	0.20	0.24	0.29	0.31	0.25	**0.86**	0.21	0.35	0.31
CA3	0.38	0.29	−0.12	0.33	0.17	0.15	0.33	0.34	0.34	**0.92**	0.25	0.43	0.39
CA4	0.33	0.26	−0.06	0.27	0.13	0.12	0.38	0.33	0.36	**0.85**	0.30	0.47	0.40
PP1	0.28	0.31	−0.13	0.23	0.18	0.25	0.43	0.46	0.41	0.25	**0.97**	0.36	0.31
PP2	0.26	0.29	−0.16	0.20	0.25	0.23	0.44	0.43	0.39	0.31	**0.92**	0.33	0.30
PP3	0.28	0.30	−0.11	0.23	0.17	0.18	0.46	0.46	0.43	0.25	**0.96**	0.35	0.30
RET1	0.42	0.33	−0.22	0.14	0.19	0.16	0.47	0.36	0.35	0.45	0.38	**0.86**	0.51
RET2	0.35	0.30	−0.09	0.23	0.17	0.10	0.42	0.29	0.36	0.42	0.29	**0.93**	0.50
RET3	0.37	0.34	−0.11	0.22	0.19	0.15	0.41	0.30	0.36	0.41	0.33	**0.90**	0.51
RET4	0.27	0.29	0.07	0.30	0.02	−0.03	0.37	0.23	0.28	0.32	0.24	**0.73**	0.39
RET5	0.39	0.36	0.07	0.37	0.14	0.11	0.45	0.26	0.28	0.42	0.31	**0.85**	0.41
SU1	0.17	0.25	−0.26	0.12	0.10	0.26	0.23	0.29	0.38	0.34	0.27	0.47	**0.79**
SU2	0.23	0.31	−0.22	0.18	0.10	0.21	0.25	0.31	0.36	0.29	0.25	0.51	**0.84**
SU3	0.23	0.18	0.01	0.11	0.11	0.11	0.10	0.19	0.34	0.23	0.15	0.27	**0.59**
SU5	0.26	0.24	−0.14	0.23	0.12	0.23	0.38	0.34	0.37	0.33	0.25	0.35	**0.72**

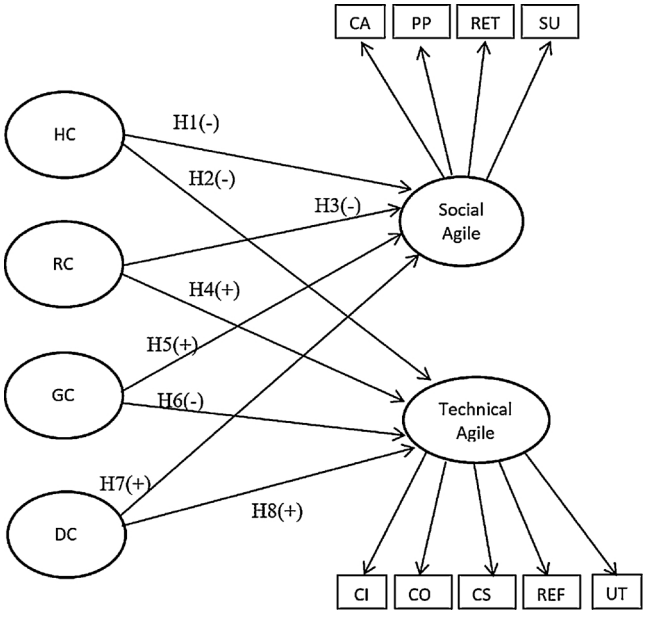

Fig. 1. Second-order (hierarchical) model with hypotheses.
Notes: HC = Hierarchical Culture, RC = Rational Culture, GC = Group Culture, and DC = Developmental Culture.

Table 9
Reliability and validity criteria of the second-order model.

	α	AVE	DC	GC	HC	RC	Social	Technical
DC	0.72	0.63	*0.79*					
GC	0.66	0.60	0.66	*0.78*				
HC	0.73	0.77	−0.26	−0.23	*0.88*			
RC	0.60	0.52	0.46	0.31	0.22	*0.72*		
Social	0.70	0.53	0.50	0.46	−0.20	0.34	*0.73*	
Technical	0.71	0.53	0.49	0.37	−0.34	0.23	0.64	*0.73*

construct in the second-order model was greater than its correlation with any other construct. In sum, like the first-order model, the second-order model demonstrated good psychometric properties.

5.2. Hypothesis testing results

We tested the significance levels of the proposed hypotheses using a bootstrapping procedure with 500 resamples (Hair & Hult, 2016). One-tailed *t* tests were used because of the directional hypotheses. The overall model accounted for 31.6% in social agile practices construct and 26.7% in technical agile practices construct. Of the eight hypotheses that we proposed in this study, five were supported (see Table 10). We predicted that hierarchical culture would be negatively related to both social (H1) and technical agile practices (H2). Both of these hypotheses were supported. Hierarchical culture tends to be

Table 10
Summary of Results.

Hypotheses		Results	Significance
H1	Hierarchical culture will have a negative impact on social agile practices usage.	Supported	$\beta = -0.14^{*}$
H2	Hierarchical culture will have a negative impact on technical agile practices usage.	Supported	$\beta = -0.27^{***}$
H3	Rational culture will have a negative impact on social agile practices usage.	*Not Supported*	$\beta = 0.20^{**}$
H4	Rational culture will have a positive impact on technical agile practices usage.	*Not Supported*	$\beta = 0.11$
H5	Group culture will have a positive impact on social agile practices usage.	Supported	$\beta = 0.24^{*}$
H6	Group culture will have a negative impact on technical agile practices usage.	*Not Supported*	$\beta = 0.07$
H7	Developmental culture will have a positive impact on social agile practices usage.	Supported	$\beta = 0.21^{*}$
H8	Developmental culture will have a positive impact on technical agile practices usage.	Supported	$\beta = 0.28^{***}$

Notes: * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$.

internally-focused and highly formalized. This culture is more concerned about the internal functioning of the organization and is less concerned about the activities of rival firms. Given that technical agile practices enable the IT departments of organizations to be well-equipped against uncertainties caused by new product (or feature) release by the rival firms, hierarchical culture, due to its internal focus, will be less likely to see a value in using technical agile practices in its software development process. Moreover, due to a stable structure where the leadership style is top-down and communication is controlled, the likelihood of using social agile practices in the IT departments with a hierarchical culture is low.

We proposed a negative relationship between rational culture and social agile practices (H3) and a positive relationship between rational culture and technical agile practices (H4). While both H3 and H4 were not supported, we actually found a significant positive relationship between rational culture and social agile practices usage, that is the opposite of what we had hypothesized. Though rational culture promotes stable structure, Iivari and Huisman (2007) argue that, due to its market-oriented focus, rational culture will likely implement new software development methodologies if their benefits are apparent. Since social agile practices, unlike traditional software development, allow everyday communication between business and technical staff, the benefits of using these practices become apparent relatively quickly. Thus, it is possible that, despite controlled structure, rationally-oriented IT departments may still encourage the use of social agile practices.

We proposed that group culture, which lies at the intersection of internal focus and flexibility, would positively relate to social agile practices (H5) but negatively relate to technical agile practices (H6). H5 was supported, whereas H6 was not supported. For H5, we had argued that group culture would encourage the use of social agile practices because of its people-oriented focus. For H6, we also suggested that group culture is less concerned about the threats posed by external markets, and as a consequence, would be less likely to promote the use of technical agile practices in IT departments. However, we did not find support for this argument.

We hypothesized that developmental culture would have a positive impact on both the use of social (H7) and technical agile practices (H8). Both H7 and H8 were supported. As discussed previously, developmental culture lies at the intersection of external focus and flexibility. While external focus is likely to encourage the use of engineering-based technical practices, flexibility in structure stimulates employee interactions and member participation, thereby supporting the use of social agile practices.

6. Discussion

The use of ASD practices has become pervasive such that IT departments in organizations in different industries and of all sizes have been actively adopting ASD practices in their software development efforts. The purpose of this study was to investigate the relationship between different cultural forms at the IT department level and ASD practices usage. We next discuss the implications of our work.

6.1. Implications

This study contributes to the emerging literature on the relationship between culture and ASD. Culture has been suggested as a critical factor affecting the adoption of ASD practices (Fruhling & Tarrell, 2008; Nerur & Balijepally, 2007); however, not much is known about how different, especially competing, forms of the IT department's culture affect the use of ASD practices. This may be partly caused by the focus of the extant literature on emphasizing the optimal agile culture rather than understanding the relationship between different cultural forms and ASD. According to the former perspective, "if the culture is not right, then the organization cannot be agile" (Lindvall et al., 2002, p. 203). While we agree that culture is an important barrier to successful ASD implementation, the label – culture - represents a broad *multifaceted* concept. Therefore, it is important to conceptualize, and subsequently operationalize culture in terms of multiple values (or dimensions) as proposed in the management literature (Alavi, Kayworth, & Leidner, 2006). Focusing on multiple dimensions of culture helps us to explain how organizations in different industries have successfully embraced ASD across their IT departments (agilescout, 2011; Thibodeau, 2012). As such, this study expands on a recent ethnographic work of Robinson and Sharp (2005a), who found ASD practices to be thriving in three fundamentally different types of organizations: (i) a large multi-national bank (ii) a medium-sized software company, and (iii) a small start-up.

From a practice perspective, the adoption of ASD in most IT departments represents a transition from the traditional plan-driven software development practices. There are norms that are defined by the traditional plan-driven software development practices. Similarly, as specified by the agile values and principles, ASD practices represent a new set of cultural assumptions. Therefore, the transition from the traditional plan-driven software development to ASD also requires a cultural transition (Highsmith, 2013). Our study results have important implications in terms of how organizations should take the appropriate perspective and mechanisms in identifying and managing the cultural compatibilities and transitions involved in the adoption process of ASD practices across their IT departments.

Before organizations deploy ASD practices, they need to understand the existing cultural profile of their IT departments and the desired cultural profile required for ASD to build the readiness that is critical for the success of ASD adoption. The six-step process proposed by Cameron and Quinn (2011) provides a practical guideline to manage the cultural transition at the department-level from the traditional software development process to ASD. First, the different stakeholders should communicate and reach agreement on the existing organizational cultural profile based on an assessment using the competing-values model. Second, the organization needs to develop a shared understanding of the desired cultural profile of its department based on the unique context and its need for ASD adoption. Third, with the understanding of the existing and the desired cultural profiles, different stakeholders should reach shared understanding and agreement about the implications of the compatibilities (or incompatibilities) between

the existing and the future cultural states. Fourth, the desired values underlying the future culture of an IT department in the post-adoption of ASD practices should be identified and agreed upon among the different stakeholders. Fifth, specific behaviors and practices that are aligned with the desired culture of an IT department should be defined and articulated to get buy-ins from the different stakeholders in order to build the readiness for the cultural transition as a part of the adoption of ASD practices. Lastly, governance and implementation plan should to be put in place to manage the actual adoption process of ASD. With an appropriate perspective of viewing the adoption of ASD as a cultural transition and a systematic approach as suggested by the above six-stage process, organizations can effectively manage the cultural changes that are critical for the successful adoption of ASD practices throughout their IT departments.

6.2. Study limitations

While the findings of this study are interesting and informative, the results should be interpreted with the study's limitations in mind. First, the sample consisted of only the US-based firms. Given that there are considerable cultural and management differences between American and non-American firms (Hofstede, 1993), caution must be exercised while making generalizations of the findings from this study to non US-based firms. Cultural differences become particularly important in the context of IT outsourcing where the clients and the vendors may be from different countries (Nidhra, Yanamadala, Afzal, & Torkar, 2013; Zahedi, Shahin, & Babar, 2016). Second, following the seminal work of Iivari and Huisman (2007), this study used a Likert-type agreement scale to assess the four cultural types proposed by the competing-values model. However, there is another way of measuring the competing-values model that allows respondents to allocate 100 points among the four cultural types (Cameron & Quinn, 2011). Using the alternative measurement approach may yield further insights. Third, although Harman's single-factor test did not provide any evidence of mono-method bias, due to the length of survey, we could not include questions for a marker variable, which could have been used to perform an alternative test of the common method bias (Malhotra, Kim, & Patil, 2006). Fourth, this study relied on LinkedIn to identify a potential pool of participants. Since people make voluntary choices on whether or not to be included in a particular community on LinkedIn, there might exist self-selection bias. However, LinkedIn has been regularly used by scholars as a platform to gain access to experienced industry professionals, who otherwise are difficult to reach (Gupta & George, 2016). Fifth, we hypothesized that IT department culture affects ASD practices. However, while old culture may constrain ASD adoption, a new culture may emerge because of ASD adoption. Our study did not capture the extent to which ASD practices change the culture over time. Lastly, while most organizations tend to apply agile practices throughout their IT department (Rigby, Sutherland, & Takeuchi, 2016), some IT departments may only adopt agile practices for few IT projects.

6.3. Future research

Future research may extend the present study by including a broader sample of organizations from different countries. In particular, an interesting avenue for future research is to explore how different cultural forms affect the use of ASD practices in the context of IT outsourcing where the clients and the vendors are from different counties. The vendor's use of ASD practices may be influenced by not only the existence of their own competing cultural forms within their organization but also by the client's culture. Additional research with data from multiple sources, especially non-LinkedIn sources, may provide further rigor and findings that complement this study's results. Given that ASD adoption is a dynamic process in which old cultures constrain the new ASD practices and at the same time, old cultures may transform into new cultures because of the ASD practices. A fruitful area for future research is use longitudinal field studies to examine how old cultures influence ASD practices, and how ASD practices change the cultures over time.

References

agilescout (2011). *Best way to set up your agile office – Is an open office right?* 2013.

Alavi, M., Kayworth, T. R., & Leidner, D. E. (2006). An empirical examination of the influence of organizational culture on knowledge management practices. *Journal of Management Information Systems, 22*(3), 191–224.

Asnawi, A. L., Gravell, A. M., & Wills, G. B. (2011). *Empirical investigation on agile methods usage: Issues identified from early adopters in Malaysia Agile Processes in Software Engineering and Extreme Programming.* Springer192–207.

Barney, J. B. (1986). Organizational culture: Can it be a source of sustained competitive advantage? *The Academy of Management Review, 11*(3), 656–665.

Beck, K. (1999). Embracing change with extreme programming. *Computer, 32*(10), 70–77.

Beck, K. (2006). *Extreme programming explained. Embrace change*.

Bellini, C. G. P., Pereira, R. D. C. D. F., & Becker, J. L. (2008). Measurement in software engineering: From the roadmap to the crossroads. *International Journal of Software Engineering and Knowledge Engineering, 18*(01), 37–64.

Berente, N., & Lyytinen, K. (2007). What is being iterated? Reflections on iteration in information system engineering processes. *Conceptual Modelling in Information Systems Engineering,* 261–278.

Berger, H. (2007). Agile development in a bureaucratic arena—A case study experience. *International Journal of Information Management, 27*(6), 386–396.

Cameron, K. S., & Quinn, R. E. (2011). *Diagnosing and changing organizational culture: Based on the competing values framework.* Jossey-Bass.

Cameron, K., Quinn, R. E., DeGraff, J., & Thakor, A. V. (2014). *Competing values leadership.* Edward Elgar Publishing.

Cao, L., Mohan, K., Xu, P., & Ramesh, B. (2009). A framework for adapting agile development methodologies. *European Journal of Information Systems, 18*(4), 332–343.

Chow, T., & Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. *The Journal of Systems and Software, 81*(6), 961–971.

Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. *Computer, 34*(11), 131–133.

Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research, 20*(3), 329–354.

Conboy, K., & Fitzgerald, B. (2010). Method and developer characteristics for effective agile method tailoring: A study of XP expert opinion. *ACM Transactions on Software Engineering and Methodology (TOSEM), 20*(1), 2.

Corvera Charaf, M., Rosenkranz, C., & Holten, R. (2013). The emergence of shared understanding: Applying functional pragmatics to study the requirements development process. *Information Systems Journal, 23*(2), 115–135.

Derby, E. (2006). *Observations on corporate culture and agile methods adoption/adaptation.* Retrieved fromhttp://www.estherderby.com/weblog/archive/2006_01_01_archive.html.

Desouza, K. C., & Evaristo, J. R. (2006). Project management offices: A case of knowledge-based archetypes. *International Journal of Information Management, 26*(5), 414–423.

Diegmann, P., & Rosenkranz, C. (2016). *Improving software development efficiency–How diversity and collective intelligence shape agile team efficiency*.

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *The Journal of Systems and Software, 85*(6), 1213–1221.

Dybå, T., & Dingsoyr, T. (2009). What do we know about agile software development? *IEEE/ACM International Conference on Automated Software Engineering Workshops IEEE/ACM International Conference on Automated Software Engineering, 26*(5), 6–9.

Feldman, M. S., & March, J. G. (1981). Information in organizations as signal and symbol. *Administrative Science Quarterly,* 171–186.

Fornell, C., & Larcker, D. F. (1981). Evaluating structural equation models with unobservable variables and measurement error. *Journal of Marketing Research,* 39–50.

Fruhling, A. L., & Tarrell, A. E. (2008). *Best practices for implementing agile methods: A guide for DOD software developers.* IBM Center for the Business of Government.

Gupta, M., & George, J. F. (2016). *Toward the development of a big data analytics capability. Information & Management*.

Hair, J. F., Jr, & Hult, G. T. M. (2016). *A primer on partial least squares structural equation modeling (PLS-SEM).* Sage Publications.

Hair, J. F., Black, W. C., Babin, B. J., Anderson, R. E., & Tatham, R. L. (2006). *Multivariate data analysis, Vol. 6*. Upper Saddle River, NJ: Pearson Prentice Hall.

Hair, J. F., Jr, Hult, G. T. M., Ringle, C., & Sarstedt, M. (2013). *A primer on partial least squares structural equation modeling (PLS-SEM).* SAGE Publications Incorporated.

Hallinger, P., & Snidvongs, K. (2008). Educating leaders: is there anything to learn from business management? *Educational Management Administration & Leadership, 36*(1), 9–31.

Harvie, D. P., & Agah, A. (2016). Targeted scrum: Applying mission command to agile software development. *IEEE Transactions on Software Engineering, 42*(5), 476–489.

Highsmith, J. (2013). *Adaptive software development: A collaborative approach to managing complex systems.* Addison-Wesley.

Hofstede, G. (1993). Cultural constraints in management theories. *Academy of Management Executive, 7*(1), 81–94.

Hummel, M. (2013). *Measuring the impact of communication in agile development: A research model and pilot test*.

Hummel, M., Rosenkranz, C., & Holten, R. (2015). The role of social agile practices for direct and indirect communication in information systems development teams. *Communications of the Association for Information Systems, 36*(1), 15.

Hung, Y. W., Hsu, S.-C., Su, Z.-Y., & Huang, H.-H. (2014). Countering user risk in information system development projects. *International Journal of Information Management, 34*(4), 533–545.
Iansiti, M., & MacCormack, A. (1996). Developing products on Internet time. *Harvard Business Review, 75*(5), 108–117.
Iivari, J., & Huisman, M. (2007). The relationship between organizational culture and the deployment of systems development methodologies. *MIS Quarterly, 31*(1), 35–58.
Iivari, J., & Iivari, N. (2011). Varieties of user-centredness: An analysis of four systems development methods. *Information Systems Journal, 21*(2), 125–153.
Iversen, J., & Ngwenyama, O. (2006). Problems in measuring effectiveness in software process improvement: A longitudinal study of organizational change at Danske Data. *International Journal of Information Management, 26*(1), 30–43.
Jarvis, C. B., MacKenzie, S. B., & Podsakoff, P. M. (2003). A critical review of construct indicators and measurement model misspecification in marketing and consumer research. *The Journal of Consumer Research, 30*(2), 199–218.
Jyothi, V. E., & Rao, K. N. (2011). Effective implementation of agile practices. *IJACSA) International Journal of Advanced Computer Science and Applications, 2*(3).
Larson, D., & Chang, V. (2016). A review and future direction of agile, business intelligence, analytics and data science. *International Journal of Information Management, 36*(5), 700–710.
Law, A., & Charron, R. (2005). *Effects of agile practices on social factors. Paper presented at the ACM SIGSOFT Software Engineering Notes*.
Leidner, D. E., & Kayworth, T. (2006). Review: A review of culture in information systems research: Toward a theory of information technology culture conflict. *MIS Quarterly, 30*(2), 357–399.
Lin, T.-C., Chen, C.-M., Hsu, J. S.-C., & Fu, T.-W. (2015). The impact of team knowledge on problem solving competence in information systems development team. *International Journal of Project Management, 33*(8), 1692–1703.
Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., et al. (2002). *Empirical findings in agile methods extreme programming and agile methods—XP/Agile universe 2002.* Springer197–207.
MacKenzie, S. B., Podsakoff, P. M., & Podsakoff, N. P. (2011). Construct measurement and validation procedures in MIS and behavioral research: Integrating new and existing techniques. *MIS Quarterly, 35*(2), 293–334.
Malhotra, N. K., Kim, S. S., & Patil, A. (2006). Common method variance in IS research: A comparison of alternative approaches and a reanalysis of past research. *Management Science, 52*(12), 1865–1883.
Maruping, L. M., Venkatesh, V., & Agarwal, R. (2009). A control theory perspective on agile methodology use and changing user requirements. *Information Systems Research, 20*(3), 377–399.
McHugh, O., Conboy, K., & Lang, M. (2011). *Using Agile practices to build trust in an Agile team: A case study Information Systems Development.* Springer503–516.
Mishra, D., Mishra, A., & Ostrovska, S. (2012). Impact of physical ambiance on communication, collaboration and coordination in agile software development: An empirical evaluation. *Information and Software Technology, 54*(10), 1067–1078.
Mnkandla, E., & Dwolatzky, B. (2007). Agile methodologies selection toolbox. *Paper Presented at the Software Engineering Advances, 2007*.
Nerur, S., & Balijepally, V. (2007). Theoretical reflections on agile development methodologies. *Communications of the ACM, 50*(3), 79–83.
Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM, 48*(5), 72–78.
Nidhra, S., Yanamadala, M., Afzal, W., & Torkar, R. (2013). Knowledge transfer challenges and mitigation strategies in global software development—A systematic literature review and industrial validation. *International Journal of Information Management, 33*(2), 333–355.
Ouchi, W. G., & Johnson, J. B. (1978). Types of organizational control and their relationship to emotional well being. *Administrative Science Quarterly,* 293–317.
Ozcan-Top, O., & Demirörs, O. (2013). *Assessment of agile maturity models: A multiple case study. Paper Presented at the International Conference on Software Process Improvement and Capability Determination*.
Park, J.-G., & Lee, J. (2014). Knowledge sharing in information systems development projects: Explicating the role of dependence and trust. *International Journal of Project Management, 32*(1), 153–165.
Pedrycz, W., Russo, B., & Succi, G. (2011). A model of job satisfaction for collaborative development processes. *The Journal of Systems and Software, 84*(5), 739–752.
Petter, S., Straub, D., & Rai, A. (2007). Specifying formative constructs in information systems research. *MIS Quarterly,* 623–656.
Podsakoff, P. M., & Organ, D. W. (1986). Self-reports in organizational research: Problems and prospects. *Journal of Management, 12*(4), 531–544.
Quinn, R. E., & Rohrbaugh, J. (1983). A spatial model of effectiveness criteria: Towards a competing values approach to organizational analysis. *Management Science, 29*(3), 363–377.
Recker, J., Holten, R., Hummel, M., & Rosenkranz, C. (2017). How agile practices impact customer responsiveness and development success: A field study. *Project Management Journal, 48*(2), 99–121.
Rigby, D. K., Sutherland, J., & Takeuchi, H. (2016). Embracing agile. *Harvard Business Review, 94*(5), 40–50.
Robinson, H., & Sharp, H. (2005a). *Organisational culture and XP: Three case studies. Paper Presented at the Agile Conference, 2005. Proceedings*.
Robinson, H., & Sharp, H. (2005b). *The social side of technical practices extreme programming and agile processes in software engineering.* Springer100–108.
Salge, C., & Berente, N. (2014). *Pair programming: A contingency approach*.
Schwaber, K., & Beedle, M. (2002). *Agile software development with scrum, Vol. 1*. Prentice Hall Upper Saddle River.
Siakas, K. V., & Siakas, E. (2007). The agile professional culture: A source of agile quality. *Software Process Improvement and Practice, 12*(6), 597–610.
So, C., & Scholl, W. (2009). *Perceptive agile measurement: New instruments for quantitative studies in the pursuit of the social-psychological effect of agile practices Agile Processes in Software Engineering and Extreme Programming.* Springer83–93.
Stavru, S. (2014). A critical examination of recent industrial surveys on agile method usage. *The Journal of Systems and Software, 94*, 87–97.
Storey, M.-A., Zagalsky, A., Singer, L., & German, D. (2015). How social and communication channels shape and challenge a participatory culture in software development. *IEEE Transactions on Software Engineering, 41*(99).
Strode, D. E., Huff, S. L., & Tretiakov, A. (2009). *The impact of organizational culture on agile method use. Paper Presented at the System Sciences, 2009* HICSS'09. 42nd Hawaii International Conference on.
Sutherland, J., & Schwaber, K. (2007). *The Scrum Papers: Scrum.*.
Thibodeau, P. (2012). *John Deere plows into agile.* 2013.
Thummadi, B. V., Shiv, O., Berente, N., & Lyytinen, K. (2011). *Enacted software development routines based on waterfall and agile software methods: Socio-technical event sequence study. Service-oriented perspectives in design science research*. Springer207–222.
Tolfo, C., & Wazlawick, R. S. (2008). The influence of organizational culture on the adoption of extreme programming. *The Journal of Systems and Software, 81*(11), 1955–1967.
Tolfo, C., Wazlawick, R. S., Ferreira, M. G. G., & Forcellini, F. A. (2011). Agile methods and organizational culture: Reflections about cultural levels. *Journal of Software Maintenance and Evolution Research and Practice, 23*(6), 423–441.
Treude, C., & Storey, M.-A. (2009). How tagging helps bridge the gap between social and technical aspects in software development. *Paper Presented at the Software Engineering, 2009*.
Tripp, J. F., & Armstrong, D. J. (2014). Exploring the relationship between organizational adoption motives and the tailoring of agile methods. *Paper Presented at the System Sciences (HICSS)*.
Vavpotic, D., & Bajec, M. (2009). An approach for concurrent evaluation of technical and social aspects of software development methodologies. *Information and Software Technology, 51*(2), 528–545.
VersionOne. (2015). 9th annual state of agile survey. Retrieved from.
Vick, T. E., Nagano, M. S., & Popadiuk, S. (2015). Information culture and its influences in knowledge creation: Evidence from university teams engaged in collaborative innovation projects. *International Journal of Information Management, 35*(3), 292–298.
vom Brocke, J., Zelt, S., & Schmiedel, T. (2016). On the role of context in business process management. *International Journal of Information Management, 36*(3), 486–495.
Weber, M. (1947). *The theory of economic and social organization. Trans. AM Henderson and Talcott Parsons.* New York: Oxford University Press.
Wetzels, M., Odekerken-Schröder, G., & Van Oppen, C. (2009). Using PLS path modeling for assessing hierarchical construct models: Guidelines and empirical illustration. *MIS Quarterly,* 177–195.
Whitworth, E., & Biddle, R. (2007). *The social nature of agile teams. Paper Presented at the Agile Conference (AGILE), 2007*.
Williams, L., & Kessler, R. (2002). *Pair programming illuminated.* Addison-Wesley Longman Publishing Co., Inc.
Williamson, O. E. (1981). The economics of organization: The transaction cost approach. *The American Journal of Sociology,* 548–577.
Zahedi, M., Shahin, M., & Babar, M. A. (2016). A systematic review of knowledge sharing challenges and practices in global software development. *International Journal of Information Management, 36*(6), 995–1019.

Manjul Gupta is an Assistant Professor in the College of Business in the Department of Information Systems and Business Analytics at Florida International University. He earned his PhD in Business Administration, majoring in Management Information Systems, from Iowa State University. His research has appeared in Management Information Systems Quarterly, Information & Management, and Communications of AIS journals.

Joey F. George is a professor of information systems and the John D. De Vries Endowed Chair in Business in the College of Business at Iowa State University. His research interests focus on the use of information systems in the workplace, including deceptive computer-mediated communication, computer-based monitoring, and group support systems.

Weidong Xia is an Associate Professor in the College of Business in the Department of Information Systems and Business Analytics at Florida International University.