Acceptance of Agile methodologies: A critical review and conceptual framework

Article *in* Decision Support Systems · March 2009

DOI: 10.1016/j.dss.2008.11.009 · Source: DBLP

2 authors:

Frank K. Y. Chan
ESSEC - Ecole Supérieur des Sciences Economiques et Commerciales
18 PUBLICATIONS **3,442** CITATIONS

James Y. L. Thong
Hong Kong University of Science and Technology
58 PUBLICATIONS **31,362** CITATIONS

Decision Support Systems 46 (2009) 803–814

Contents lists available at ScienceDirect

Decision Support Systems

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

Acceptance of agile methodologies: A critical review and conceptual framework

Frank K.Y. Chan, James Y.L. Thong *

Department of Information Systems, Business Statistics and Operations Management, School of Business and Management, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

ARTICLE INFO

Available online 20 November 2008

Keywords:
Systems development methodologies
Agile methodologies
Knowledge management
Framework

ABSTRACT

It is widely believed that systems development methodologies (SDMs) can help improve the software development process. Nevertheless, their deployment often encounters resistance from systems developers. Agile methodologies, the latest batch of SDMs that are most suitable in dealing with volatile business requirements, are likely to face the same challenge as they require developers to drastically change their work habits and acquire new skills. This paper addresses what can be done to overcome the challenge to agile methodologies acceptance. We provide a critical review of the extant literature on the acceptance of traditional SDMs and agile methodologies, and develop a conceptual framework for agile methodologies acceptance based on a knowledge management perspective. This framework can provide guidance for future research into acceptance of agile methodologies, and has implications for practitioners concerned with the effective deployment of agile methodologies.

1. Introduction

A systems development methodology (SDM), defined as a documented collection of policies, processes, and procedures, is commonly used by software development teams to improve the software development process in terms of increased productivity of information technology (IT) personnel and higher quality of the final IT solutions [40,64]. Some examples of traditional SDMs include structured modeling-based methodology and workflow development methodology (e.g., [63,90]). SDMs are constantly evolving to keep up with changing technologies and satisfy new demands from users. More recently, a new generation of SDMs called agile methodologies is claimed to be better suited for dealing with the dynamic business environment [47]. Agile methodologies, such as Extreme Programming (XP) and Scrum, have received praise from practitioners because of their abilities to deal with volatile requirements [66].

Although SDMs are believed to help improve the software development process, their deployment often encounters resistance from software developers. Only about half of all organizations actually follow an SDM diligently [27]. Organizations attempting to deploy an SDM often encounter resistance from their IT personnel [48,62]. In the case of agile methodologies, their deployment is also similarly resisted. Industry surveys (e.g., [5,7]) have indicated that the acceptance of agile methodologies in organizations is still at an early phase, with many respondents indicating a limited knowledge of agile methodologies. In the case of organizations that have employed traditional SDMs for years, the migration to agile methodologies also poses challenges, such as a change in management style, changes to the systems development process, need for tighter collaboration between project members, and compatibility of technologies [59]. Thus, similar to the situation faced by traditional SDMs [69], overcoming resistance to agile methodologies acceptance is a critical area of concern in Information Systems (IS) research.

There are significant differences between traditional SDMs and agile methodologies. Agile development emphasizes the people factor [16]—both software developers and users play important roles in agile development. In traditional SDMs, users participate mainly in specification development, with minimal involvement in other activities [59]. However, in agile development, customers (who represent system users) work in small teams with developers as active team members. For example, customers and developers jointly determine the system features to be implemented in each development cycle. Such a drastic change in the user's role suggests that successful acceptance of agile methodology is concerned not only with software developers and organizations, but also with customers who are expected to be collaborative, representative, authorized, committed, and knowledgeable [9]. The relationship with customers is likely to be an important consideration when software developers decide whether to adopt agile methodologies, as it is not easy to find such "qualified" customers, especially for complex systems [59]. Due to the differences between traditional SDMs and agile methodologies, the previously-identified facilitators and barriers to acceptance of traditional SDMs may not apply to the acceptance of agile methodologies, and there may be other factors that are more relevant in this context.

* Corresponding author. Tel.: +852 23587631; fax: +852 23582421.
E-mail addresses: frankcky@ust.hk (F.K.Y. Chan), jthong@ust.hk (J.Y.L. Thong).

0167-9236/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2008.11.009

Existing research has attempted to examine developers' acceptance of SDMs from the technology adoption perspective. These studies treat SDMs as technology innovations and make use of the technology adoption models, such as Technology Acceptance Model (TAM) (e.g., [68]). Although these models have been found to be suitable for examining the acceptance of traditional SDMs, they focus mainly on the technology characteristics, such as perceived usefulness and perceived ease of use. Non-technology factors, although they have their merits, have been largely neglected in previous research. For instance, individual characteristics and organizational characteristics can also play critical roles in the acceptance of traditional SDMs [76]. Further, there is a scarcity of empirical research in the domain of traditional SDMs, with research on agile methodologies mostly confined to case studies. Thus, there is an urgent need to conduct a critical review of the extant literature to develop a conceptual framework for agile methodologies acceptance. In developing this conceptual framework, it will be important to consider multiple aspects—notably individual, organizational, and technological characteristics—in order to arrive at a balanced and comprehensive understanding of agile methodologies acceptance.

At the same time, the deployment of SDMs has impact on knowledge management in software development. Software development is a knowledge-intensive business involving many people working in different phases and activities [73], and requires effective knowledge integration and transfer within development teams [43,55]. For agile methodologies, the management of knowledge is particularly important. For example, developers have to acquire the knowledge of using agile methodologies before they can actually use the methodologies to guide the development of software. They have to retain knowledge, such as coding standards, within their development teams, and transfer the knowledge to other team members. Further, agile methodologies change the way developers learn about the software product. For instance, agile methodologies discourage the use of formal documentation. The product knowledge becomes tacit and its transfer relies on rotation of team member in different phases throughout the project [59]. Also, the transfer of product knowledge between developers and customers may involve interorganizational knowledge transfer. These issues highlight the need to understand what factors contribute to the creation, retention, and transfer of knowledge about the agile methodology (e.g., coding standards), as well as the software product (e.g., product specification). Such knowledge is crucial to the successful acceptance of agile methodology. Thus, it is of practical and scientific significance to examine acceptance of agile methodologies from a knowledge management perspective.

Whetten [87] suggested that the "what," "how," and "why" are three essential elements of theories. We follow Whetten's [87] principle of theory development to guide our literature review and the development of our conceptual framework. First, we conduct a critical analysis of a comprehensive review of the extant literature on traditional SDMs and agile methodologies with the purpose of identifying potential factors that are critical to the software developer's acceptance of agile methodologies. This step helps to identify the "what" elements of theory development. Next, we adopt a fresh theoretical perspective—specifically knowledge management—in discussing the effects of the potential factors on acceptance of agile methodologies. This step assists in explaining the "how" and "why" the potential factors influence acceptance of agile methodologies. This critical analysis is organized along the following research questions: (1) What are the different types of characteristics or factors that can influence a developer's acceptance of agile methodologies? (2) From the knowledge management perspective, how do these characteristics influence the developer's acceptance of agile methodologies?

This paper makes the following contributions to research and practice. First, we synthesize existing knowledge from prior empirical studies on the acceptance of traditional SDMs and studies on agile methodologies. The factors identified from the extant literature are then critically examined to determine their applicability to the domain of agile methodologies. Second, we develop a conceptual framework based on the knowledge management perspective and discuss the effects of various factors on acceptance of agile methodologies. Third, we offer suggestions on how the conceptual framework can drive future research. Finally, we provide advice to IS practitioners concerning the effective deployment of agile methodologies based on the existing knowledge.

In the following sections, we first give a brief introduction to SDMs and agile methodologies. Next, we review prior studies on the acceptance of traditional SDMs and agile methodologies covering a period from the late 1990s, when researchers started to examine acceptance of traditional SDMs and agile methodologies were first introduced, to the present. To our knowledge, there are only a limited number of empirical studies concerning the acceptance of traditional SDMs, and no empirical studies have been conducted in the domain of agile methodologies. We then incorporate the findings from the review into a conceptual framework for acceptance of agile methodologies. Finally, we discuss the contributions to research and practice.

2. Background

2.1. Barriers to acceptance of SDMs

Previous research suggests that SDMs help improve the process and product of systems development. For example, the use of SDMs helps subdivide the complex process of systems development into plausible and coherent steps, facilitate management of the development process, and facilitate acquisition and systematic storage of knowledge [25]. Despite the potential advantages of using SDMs, practitioners have been slow in adopting SDMs. Fitzgerald [25] found that 60% of organizations surveyed were not using methodologies, whereas only 6% followed a methodology rigorously, and 79% of those not using any methodology did not intend to adopt one. Some possible reasons for the low acceptance are: (1) developers simply ignore the newly introduced SDMs, (2) SDMs treat the process of systems development as an orderly rational process when it is not, and (3) SDMs are assumed to be universally applicable, when they should be adjusted across different development situations [25]. In summary, the acceptance of SDMs among practitioners remains a perennial challenge that attracts researchers' interest [89].

2.2. Agile methodologies versus traditional SDMs

Agile methodologies are a new host of methodologies that claim to overcome the limitations of traditional plan-driven SDMs. The "Agile Manifesto" published by a group of software practitioners outlines the principles of agile systems development [50]. In short, these principles emphasize the importance of individuals and their interactions, customer collaboration, early and continuous delivery of software, and the capability to respond to volatile requirements. Examples of agile methodologies that align with the Agile Manifesto include Extreme Programming (XP), Crystal methods, Lean Development, Scrum, and Adaptive Software Development (see [1,33]).

Highsmith [33] suggested that the differences between traditional SDMs and agile methodologies rest on two assumptions about customers. First, traditional SDMs assume that customers do not know their requirements but developers do, whereas agile methodologies assume that both customers and developers do not have full knowledge of system requirements at the beginning [33]. Thus, in traditional SDMs, developers want a detailed specification, in order to absolve themselves of responsibility by claiming that they just build the system in a way specified by the customer, whereas in agile methodologies, both customers and developers learn about the

system requirements as the development process evolves, without recrimination [33]. Second, traditional SDMs assume customers are short-sighted, and thus developers have to build in extra functionalities to meet the future needs of customers, often leading to over-designed system [33]. On the other hand, agile methodologies emphasize simplicity—the art of maximizing the work not done [50]. Also, the differences in philosophy between traditional SDMs and agile methodologies lead to differences in a number of practices and requirements, such as planning and control, role assignment among developers, customer's role, and technology used [59].

The above differences bring a number of challenges to a successful transition to agile methodologies. Nerur et al. [59] suggested that migrating to agile methodologies involved many issues pertaining to management, people, process, and technology. The complications in migrating to agile methodologies make it uncertain that prior findings on the acceptance of traditional SDMs would be readily applicable to agile methodologies. For example, factors previously found to be significant for traditional SDMs may be inadequate in capturing the distinct characteristics (e.g., frequent releases, and continuous improvement) and usage contexts (e.g., rapidly changing requirements, and face-to-face conversation) of agile methodologies. Further, prior studies on SDMs (e.g., [29,30,68]) focus solely on developers' beliefs about an SDM (e.g., perceived usefulness and perceived ease of use) and fail to take into account the importance of people-related issues (e.g., individual competence) and management-related issues (e.g., management style). Hence, it is imperative to incorporate different aspects in understanding acceptance of agile methodologies [1].

In the next section, we review the prior literature on SDMs and agile methodologies.

3. Review of prior literature

3.1. Identification of the relevant literature base

We first assess the current state of knowledge with respect to acceptance of traditional SDMs, such as Structured Model, Waterfall Model, and Object-oriented Systems Development (OOSD). In particular, OOSD is occasionally regarded as a programming language instead of an SDM. However, Fichman and Kemerer [23] suggested that OOSD "is not just a language generation or a database model, but the entire procedural paradigm for software development," and it qualifies as a process innovation. Thus, OOSD is regarded as an SDM for the purpose of our study and prior research on OOSD is included in our scope of review. On the other hand, studies on the acceptance of systems development tools and techniques, such as CASE and structured programming, are outside of our scope of review. The reason is that systems development tools and techniques are used within and are components of the systems development process [85]. Also, acceptance of a systems development process represents a much more radical change than adopting tools or techniques [69]. As a result, factors that are important to the adoption of tools or techniques may not be applicable to the domain of SDMs.

The second part of our review focuses on studies of agile methodologies. Although there are now more studies on agile methodologies, they tend to focus on the practical usage of agile methodologies (e.g., [8]) and their costs and benefits (e.g., [61]), with no empirical study on the factors affecting the acceptance of agile methodologies. In a review on agile methodologies, Erickson et al. [22] reported similar findings that previous research mainly consisted of case studies, comparative analysis and experience reports, with very few empirical studies. In total, we identified seven articles that empirically examined individual acceptance of traditional SDMs [29,30,42,45,68,76,79], three articles that empirically examined organizational acceptance of SDMs [15,24,32], four articles that aimed to identify important factors in the implementation of SDMs [41,69–71], and eight case studies that discussed the acceptance of agile methodologies [13,16,18,21,34,53,59,74]. The lack of empirical studies of agile methodologies highlights the need to develop a conceptual framework for the acceptance of agile methodologies, which can then form a basis for future empirical research.

3.2. Acceptance of SDMs

Table 1 summarizes the prior studies on acceptance of traditional SDMs. Existing theories for examining the acceptance of Information Technology (IT) tools (e.g., Innovation Diffusion Theory [72]; Technology Acceptance Model (TAM; [20]) have been widely used for examining individual intentions to adopt IT innovations, such as the World Wide Web (e.g., [3]) and spreadsheets (e.g., [14]). Johnson [42] performed an initial investigation into individual acceptance of SDMs using TAM. In his study, the perceived usefulness and perceived ease of use scales were shown to be reliable measures in general in the systems development context. Riemenschneider et al. [68] further tested the applicability of five theoretical models—TAM, TAM2, Theory of Planned Behavior (TPB), Perceived Characteristics of Innovating (PCI), and the Model of Personal Computer Utilization (MPCU) in the domain of individual acceptance of SDMs. Among 11 determinants (from the aforementioned five models) of the intention to use the new SDM (which was based on the structured development paradigm), four were found to be significant—perceived usefulness, voluntariness, perceived compatibility, and subjective norm. In a related study, Hardgrave et al. [29] investigated the determinants of individual developer's intentions to follow methodologies, based on two theoretical paradigms, TAM and diffusions of innovations (DOI). In their study, perceived usefulness, social pressure, perceived compatibility and organizational mandate were found to have a direct influence on individual developers' intentions to follow methodologies, whereas social pressure, complexity and perceived compatibility were found to be significant determinants of perceived usefulness. Templeton and Byrd [79] reported similar findings that perceived ease of use and perceived compatibility were significant determinants of relative advantage, which is equivalent to perceived usefulness in TAM [56].

On the other hand, some previous studies focus on characteristics of individual developers (e.g., experience) and organizations (e.g., management support), in addition to the characteristics of SDMs. Sultan and Chan [76] reported that characteristics of the technology, i.e., relative advantage, perceived compatibility, and complexity, are not significant in differentiating adopters and non-adopters of OO technology. The results contradicted the findings in other studies (e.g., [29]), possibly due to the fact that both adopters and non-adopters are fully aware of the claimed benefits of the technology since they are experienced developers, and they may differ in their adoption decisions due to other factors, such as characteristics of individual developers and organizations [76].

Another branch of prior research focuses on the organizational factors that can influence acceptance of SDMs. Studies on the assimilation of OO technology (e.g., [15,24,69]) have suggested that learning-related factors, such as training and external support, are positively related to organizational assimilation of OO technology. Further, organizational culture has been found to be an important factor that would affect the IT managers' perceptions of SDM, as well as the extent to which mandatoriness of SDM use and social norms influence actual SDM use [41]. These studies suggest a number of organizational factors that are crucial to the successful assimilation of SDMs. However, these studies have not taken characteristics of developers or characteristics of SDMs into consideration. On the whole, these studies identified a number of organizational factors (e.g., management support, training, and organizational culture) that are potential determinants of acceptance of SDMs, and suggested some additional characteristics of SDMs (e.g., maturity of technology) that may contribute to acceptance of SDMs.

Table 1
Summary of prior studies on acceptance of traditional SDMs

Source	Methodology studied	Independent variables studied	Dependent variables
Hardgrave et al. [29]	Custom-created methodology based on the structured development paradigm	Perceived usefulness[a], complexity[a], social pressure[a], perceived compatibility[a], organizational mandate[a]	Intention to follow methodology
Hardgrave and Johnson [30]	Object-oriented systems development (OOSD)	Subjective norm[a], organizational usefulness[a], personal usefulness, perceived behavioral control (internal)[a], perceived behavioral control (external)	Intention to use new ISD process
Johnson [42]	Object-oriented systems development (OOSD)	Perceived usefulness[a], perceived ease of use[a]	Behavioral intention
Khalifa and Verner [45]	Waterfall model and prototyping	Facilitating condition (team size[a], innovation[a], organizational support), product quality (software quality, software maintainability), process quality (project control[a], communication with users[a], early detection of problem, development cost)	Use of methods
Riemenschneider et al. [68]	Custom-created methodology based on the structured development paradigm	Perceived usefulness[a], perceived ease of use, subjective norm[a], voluntariness[a], perceived compatibility[a], image, visibility, perceived behavioral control (internal), perceived behavioral control (external), career consequences, result demonstrability	Behavioral intention to adopt
Sultan and Chan [76]	Object-oriented technology	Experience, firm's values[a], teamwork[a], opinion leadership[a], communication[a], response to risk[a], company culture[a], company structure[a], centralization[a], formalization[a], integration[a], technology policy[a], competitive strategies[a], management risk perception[a], top management support[a], relative advantage, compatibility, complexity	Adoption
Templeton and Byrd [79]	Systems development methodology (SDM)	Perceived ease of use[a], perceived compatibility, triability[a], knowledge of proximity, voluntariness	Relative advantage
Cho and Kim [15]	Object-oriented technology	Expectation for market trend, maturity of technology[a], technology compatibility, relative benefits, complexity, managerial innovativeness, intensity of new technology education[a], satisfaction with existing technology[a], experience	Technology assimilation
Fichman and Kemerer [24]	Object-oriented programming language (OOPL)	Learning-related scale[a], related knowledge[a], diversity[a]	Technology assimilation
Higgins and Hogan [32]	Information engineering (IE) systems development methodology and software	Cross-functional team spirit[a], top management support[a], user participation[a], technical transfer	Perceived success of IE implementation
Iivari and Huisman [41]	Systems development methodology (SDM)	Organizational culture orientations[b], mandatoriness of SDM use[b], social norms concerning SDM use[b]	Implementation of SDMs
Roberts and Hughes [70]	Systems development methodology (SDM)	Management commitment[b], sound implementation plan[b], methodology training, good change management[b], understanding of methodology[b]	Implementation of SDMs
Roberts et al. [69]	Systems development methodology (SDM)	Understanding methodology specifics and benefits[b], system personnel manager involvement with and responsibility for organizational SDM transition[b], functional manager involvement and support[b], external support[b], use of models[b]	Implementation of SDMs
Roberts et al. [71]	SDM offered by consultant companies	External support (knowledge links, such as consultants, universities, vendor training program)[b]	Implementation of SDMs

[a] Factors that were empirically found to be significant.
[b] Factors that were suggested to be important.

3.3. Acceptance of agile methodologies

Table 2 presents a summary of previous studies on agile methodologies and their corresponding identified important factors. Most of the prior studies on agile methodologies are case studies, and there are few empirical studies which examine how these factors affect the adoption of agile methodologies. Consistent with the findings in previous studies on traditional SDMs, studies on agile methodologies suggest that individual characteristics (i.e., ability of developers) are important in deploying agile methodologies (e.g., [13,18]) and using

Table 2
Summary of previous case studies on agile methodologies

Source	Description	Methodology studied	Identified factors
Ceschi et al. [13]	The authors conducted a survey to compare and contrast agile companies and plan-based companies.	Agile (e.g., XP and Scrum)	Teamwork, individual ability, motivation
Cockburn and Highsmith [16]	The authors described the effects of working in an agile style.	Agile	Individual competence, management support, communication, compatibility of agile methods, teamwork, project type, team size
Cohn and Ford [18]	The authors described common pitfalls and effective approaches to introduce agile process into an organization.	Scrum, XP	Resistance due to past experience, micromanagement (leadership), career consequences, developers' ability
Drobka et al. [21]	The authors piloted XP with four development teams and discussed the qualitative and quantitative results of using XP.	XP	External support
Highsmith and Cockburn [34]	The authors provided an overview of agile software development.	Agile	Teamwork
McManus [53]	The author discussed the people issues in agile methods.	Agile	Individual competence, compatibility between skills and tasks, good communication skills, experience in software development
Nerur et al. [59]	The authors discussed the challenges of migrating to agile methods by contrasting them with traditional SDMs.	Agile	Organizational culture, management style, organizational form, management of software development knowledge, reward systems, teamwork, competence, customer relationships, existing technology and tools, training
Schatz and Abdelshafi [74]	The authors provided a case study of deploying scrum.	Scrum	External support, teamwork, compatibility of methods, negotiation skills

agile methodologies effectively (e.g., [16]). This may be due to the fact that agile methodologies emphasize the people factor [16].

The characteristics of agile methodologies, such as compatibility, have also been found to be important factors affecting the acceptance of agile methodologies (e.g., [16,53,74]). Moreover, Nerur et al. [59] pointed out some important organizational characteristics, such as organizational culture, teamwork, and training, consistent with the findings by Sultan and Chan [76].

One distinction between studies on traditional SDMs and studies on agile methodologies is that studies on SDMs often focus on factors associated with software developers and organizations only, whereas studies on agile methodologies need to examine factors associated with users, such as customer relationships (e.g., [13,59]). This is due to the fact that users play a more active role throughout the development process in agile development, whereas users' participation is minimal in most of the activities except specification development in traditional SDMs [59].

3.4. The need for a conceptual framework

Our critical review of the prior literature leads us to make four observations. First, prior studies illustrate that existing models of acceptance of IT tools, such as TAM, can be readily used for examining the acceptance of SDMs (e.g., [68]). These models provide well-established constructs for measuring characteristics of SDMs, such as perceived usefulness and perceived ease of use. Second, it has been shown that non-technology factors, such as individual characteristics and organizational characteristics, can also affect the acceptance of SDMs, and even dominate the characteristics of SDMs in some cases (e.g., [76]). Third, studies have emphasized the importance of organizational factors in acceptance of SDMs (e.g., [15]). Finally, case studies on agile methodologies suggest that customers play a critical role in agile development and the success of agile development will hinge on finding customers who actively participate in the development process. This suggests that factors associated with customers would be crucial to the adoption of agile methodologies. In sum, while prior studies have empirically examined some of the above factors in the context of SDMs, it is still a fragmented piece of work, with no study considering the combined influences of various characteristics. Further, prior studies have made heavy use of the technology adoption perspective, with little attention to other aspects, such as the knowledge aspect of systems development, thus hampering the development of a better understanding of the acceptance of agile methodologies.

In the next section, we describe our conceptual framework and define the constructs in it. We also discuss how these constructs are crucial to acceptance of agile methodologies.

4. Conceptual framework

Our conceptual framework is depicted in Fig. 1. As the main essence of agile methodologies involves interaction between systems developers and customers, and knowledge management issues, we develop our conceptual framework based on a knowledge management perspective. We draw on the knowledge management literature to identify factors that facilitate the knowledge management process, and key knowledge management outcomes. Argote et al. [6] provided a framework for organizing the literature on knowledge management. They suggested that successful knowledge management depended on individual's ability, motivation, and opportunity to perform. Further, they suggested that knowledge creation, retention, and transfer were the key knowledge management outcomes. Based on the framework by Argote et al. [6], we classify the factors identified from previous studies on SDMs and agile methodologies into ability-related factors, motivation-related factors, and opportunity-related factors. Then, we discuss how these factors are relevant to the knowledge management process, and lead to the key knowledge management outcomes in the software development context. In particular, knowledge management outcomes, ability-related factors, motivation-related factors, and opportunity-related factors are relevant to the knowledge management aspect of systems development in understanding individual acceptance of agile methodologies. Agile methodology characteristics serve to complement the framework by adding the long-standing technology adoption perspective.

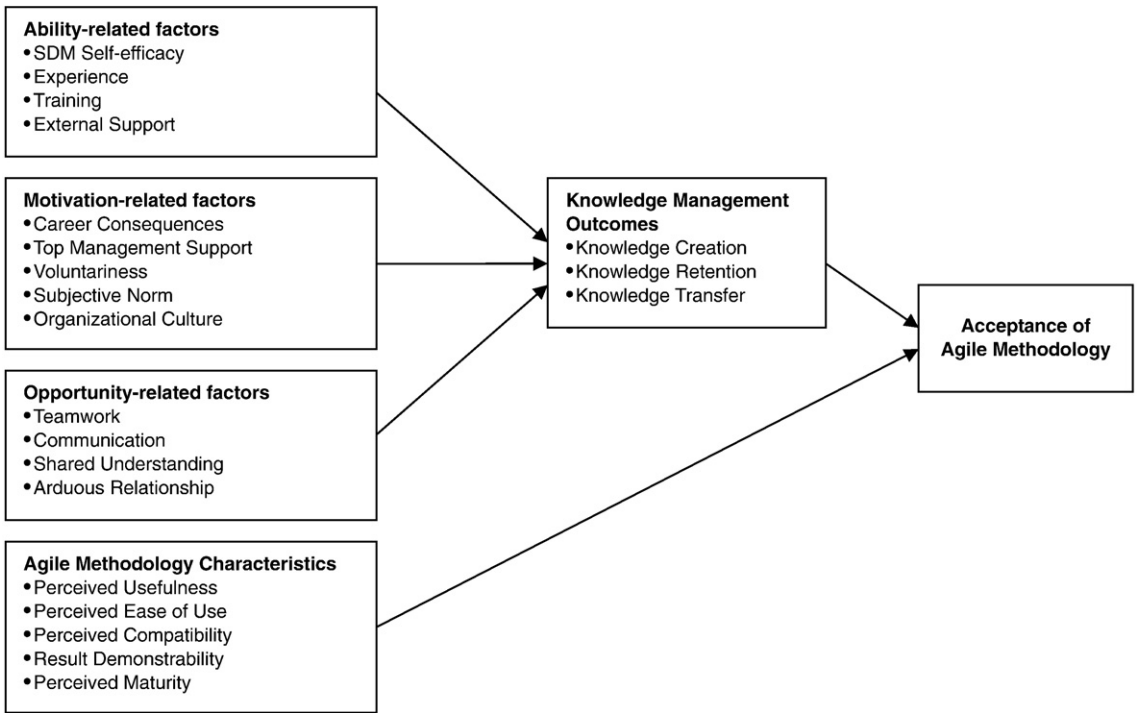

Fig. 1. Conceptual framework.

4.1. Knowledge management outcomes

Software developers have often engaged in knowledge management-related activities aimed at learning, capturing, and reusing experience [73]. To develop software, development teams have to fulfill a number of knowledge needs. For example, they have to (1) acquire knowledge about new technologies, (2) access knowledge about the domain for which the software is being developed, (3) share knowledge about local policies and practices, (4) capture knowledge and know what team members know, and (5) collaborate and share knowledge among team members [73]. Since agile methodologies emphasize individual competence, constant communication, and close collaboration between developers and customers [16], the capability to create and utilize knowledge among the development team members is critical.

To capture the capability to create and utilize knowledge among the development team members, we define knowledge management outcomes as knowledge creation, knowledge retention, and knowledge transfer following Argote et al. [6]. *Knowledge creation* refers to the generation of knowledge in organization; *knowledge retention* refers to the embedment of knowledge in a repository that persists over time; and *knowledge transfer* refers to the transfer of experience acquired in one unit to another [6]. These outcomes are inter-related. For instance, for an organization to transfer knowledge, the knowledge must be retained, whereas attempts to transfer knowledge can lead to the creation of new knowledge [6]. Further, Argote et al. [6] suggested that these outcomes were subject to the influences of properties of units, relationships between units, and knowledge.

In the domain of agile development, knowledge creation involves developing new tacit and explicit knowledge about the agile methodology (e.g., programming techniques and coding standards) and the software product (e.g., customer requirements and product specifications). As such knowledge is important to the success of agile development, it has to be retained in the development teams and transferred effectively among team members. Both knowledge retention and transfer can be achieved by agile practices such as rotation of team members in different phases throughout the project [59]. On the whole, the three knowledge management outcomes—knowledge creation, knowledge retention, and knowledge transfer—are interrelated [6] and will determine how successful the developers manage the knowledge required for using agile methodologies. Success in managing these knowledge outcomes will increase the confidence of team members in adopting agile methodologies. Developers will have the confidence to take full advantage of the agile methodologies, and the barriers (e.g., the knowledge needs suggested by Rus and Lindvall [73]) to using agile methodologies will be lowered. Thus, positive knowledge management outcomes are expected to improve individual developers' acceptance of agile methodologies.

4.2. Ability-related factors

Ability-related factors include SDM self-efficacy, experience, training, and external support (see Table 3 for definitions). These factors represent developers' abilities (i.e., SDM self-efficacy and experience), and interventions that improve developers' abilities in using agile methodologies (i.e., training and external support). Previous research has found that ability can be affected by experience, and also result from training [17,58].

SDM self-efficacy is defined as a judgment of one's capability to use an SDM [19]. Previous research has suggested that individuals with high self-efficacy may perform tasks that go beyond the specific job requirements [60]. Cabrera et al. [12] found that self-efficacy had a significant impact on people's inclination to participate in voluntary knowledge sharing. If developers believe that they have the capability to use an agile methodology, they are more likely to use it for software development, and engage in related knowledge management activities which may not be specified by the job requirements.

Further, previous studies have suggested the importance of self-efficacy in computing and high technology domains [4]. For example, self-efficacy has been found to affect computer use (e.g., [37]), early adoption of computer system (e.g., [11]), and behavioral intention (e.g., [35]). In the domain of agile methodologies, individual ability of software developers is often stressed (e.g., [13,16,18,53,59]), because if the people on the project are good enough, they can use almost any process to accomplish their tasks [16]. Thus, developers with high self-efficacy will be able to take full advantage of the agile methodology to perform their tasks, and also find it easier to use. In sum, SDM self-efficacy can encourage developers' knowledge management behaviors and improve their acceptance of agile methodologies.

Experience is defined as encounters that one undergoes or lives through [76]. In the software industry, experience can be associated with prior technical knowledge (e.g., the variety of SDMs) a software developer possesses, since prior research on software development has shown the variety of programming languages mastered by a developer to be a better indicator of knowledge and expertise than the length of experience [10]. Previous research has suggested that experience is an important factor that increases an individual's ability to manage knowledge [6]. Cohen and Levinthal [17] suggested that individuals have the capacity to understand knowledge in areas where they have previous experience because individuals learn, or absorb, knowledge by associating it with what they already know. Thus, developers with more experience are more capable of understanding and managing knowledge about the agile methodology.

Further, previous research in technology adoption has suggested that experienced individuals encounter more decisions and perceive an increase in their ability to make innovation decisions [35]. Also, agile development tends to rely on developers' knowledge of different methodologies, such as XP, Crystal, and Scrum [53]. With more experience in using various SDMs, developers will have a greater ability to recognize the advantages of the agile methodology and find it easier to use. In sum, experience can encourage developers' knowledge management behaviors and improve their acceptance of agile methodology.

Training is traditionally defined as "the formal procedures which a company utilizes to facilitate learning so that the resultant behavior contributes to the attainment of the company's goals and objectives" [51]. Previous research has suggested that abilities can result from training [58]. Training increases an individual's ability to transfer knowledge accumulated on one task to a related task [80]. Thus, training helps increase developers' abilities to learn about the agile methodology and transfer the knowledge to the real practice.

Further, since the adoption of a technological innovation creates uncertainty in the minds of potential adopters about its likely consequences [72], training is an opportunity for potential adopters to learn about an innovation and reduce the uncertainty. Then, they may be able to better evaluate the consequences of adopting the innovation. For example, Agarwal and Prasad [2] found that structured training positively affected individuals' beliefs about perceived compatibility of an innovation; Venkatesh [83] found that training was an effective intervention that helped create favorable user perceptions; Riemenschneider and Hardgrave [67] found that training had a positive effect on perceived ease of use of an SDM; Roberts and Hughes [70] suggested methodology training was the key to successful SDM implementation. In sum, training can encourage developers' knowledge management behaviors and improve their acceptance of agile methodologies.

External support refers to the use of external training and consultants [69,71]. Roberts et al. [69] suggested that outside expertise is frequently required because, in many cases, no one in the organization has the expertise required to effectively implement a new SDM. Further, internal trainers may not possess specialized skills

Table 3
Determinants of acceptance of agile methodologies

Category	Factors	Definition	Previous studies on SDMs	Previous studies on agile methodologies
Ability-related factors	SDM self-efficacy	Self-efficacy refers to a judgment of one's capability to use an SDM [19].	N/A	Ceschi et al. [13]; Cockburn and Highsmith [16]; Cohn and Ford [18]; McManus [53]; Nerur et al. [59]
	Experience	Prior technical knowledge an individual possesses, for example, the variety of SDMs, since prior research on software development has shown variety of languages to be a better indicator of knowledge and expertise than length of experience [10].	Sultan and Chan [76]	Cohn and Ford [18]; McManus [53]
	Training	Training is traditionally defined as "the formal procedures which a company utilizes to facilitate learning so that the resultant behavior contributes to the attainment of the company's goals and objectives" [51].	Cho and Kim [15]; Riemenschneider and Hardgrave [67]; Roberts and Hughes [70]	N/A
	External support	External support refers to the use of external training and consultants [69].	Roberts et al. [69]; Roberts et al. [71]	Drobka et al. [21]; Schatz and Abdelshafi [74]
Motivation-related factors	Career consequences	Career consequences, the dimension of long-term consequences of use, refer to outcomes that have a payoff in the future, such as increasing the flexibility to change jobs or increasing the opportunities for more meaningful work [81].	Riemenschneider et al. [68]	Cohn and Ford [18]
	Top management support	Top management support is the continual active and enthusiastic approval of senior executives for a proposed innovation [76].	Sultan and Chan [76]; Higgins and Hogan [32]; Roberts and Hughes [70]; Roberts et al. [69]	Cockburn and Highsmith [16]
	Voluntariness	Voluntariness is defined as "the extent to which potential adopters perceive the adoption decision to be non-mandatory" [3,31,56].	Hardgrave et al. [29]; Riemenschneider et al. [68]	N/A
	Subjective norm	Subjective norm is defined as the influence of important referents on an individual's acceptance of an SDM [30].	Hardgrave and Johnson [30]; Hardgrave et al. [29]; Riemenschneider et al. [68]	N/A
	Organizational culture	Organizational culture is defined as "a pattern of basic assumptions invented, discovered or developed by a given group as it learns to cope with its problems of external adaptation and integration that has worked well enough to be considered valid and, therefore, is to be taught to new members as the correct way to perceive, think, and feel in relation to those problems" [75].	livari and Huisman [41]	N/A
Opportunity-related factors	Teamwork	Teamwork is defined as the individual willingness to continue working together with the same team as well as in other teams [82].	Sultan and Chan [76]	Ceschi et al. [13]; Cockburn and Highsmith [16]; Highsmith and Cockburn [34]; Nerur et al. [59]; Schatz and Abdelshafi [74]
	Communication	Communication is a process by which individuals exchange information through a common system of behavior [57].	Sultan and Chan [76]	Cockburn and Highsmith [16]; McManus [53]
	Shared understanding	Shared understanding represents the extent to which the work values, norm, philosophy, problem-solving approaches, and prior work experience of a dyad are similar [26].	N/A	Ceschi et al. [13]; Nerur et al. [59]
	Arduous relationship	Arduous relationship is defined as an emotionally laborious and distant relationship between a source and a recipient [78].	N/A	Ceschi et al. [13]; Nerur et al. [59]
Agile methodology characteristics	Perceived usefulness	Perceived usefulness refers to the degree to which an individual expects that adopting an SDM will improve his or her individual job performance [29].	Hardgrave et al. [29]; Riemenschneider et al. [68]	N/A
	Perceived ease of use	Perceived ease of use is defined as the degree to which an individual believes that using a particular SDM would be free of physical and mental effort [29].	Hardgrave et al. [29]; Riemenschneider et al. [68]	N/A
	Perceived compatibility	Perceived compatibility refers to the degree to which an individual regards the practice of adopting an SDM as being consistent with his or her pre-existing software development process [29].	Cho and Kim [15]; Hardgrave et al. [29]; Riemenschneider et al. [68]	Cockburn and Highsmith [16]; McManus [53]; Schatz and Abdelshafi [74]
	Result demonstrability	Result demonstrability refers to the degree to which an innovation is perceived to be amenable to demonstration of tangible advantages [56].	Riemenschneider et al. [68]	N/A
	Perceived maturity	Perceived maturity is a function of technology uncertainty and inexperience [49].	Cho and Kim [15]	N/A

and expertise to develop training courses and materials. In the domain of agile methodologies, Droka et al. [21] suggested that a consultant who is outside an organization will bring a fresh perspective to the development team and ease the transition to agile methodologies. Similarly, Schatz and Abdelshafi [74] suggested that it will be extremely helpful to get honest, objective feedback from an outside source. Thus, external support is likely to have a similar influence as training on developers' knowledge management behaviors and their acceptance of agile methodology.

4.3. Motivation-related factors

Motivation-related factors include career consequence, top management support, voluntariness, subjective norm, and organizational

culture (see Table 3 for definitions). Career consequences and top management support represent the incentives that motivate developers to use agile methodologies, whereas voluntariness, subjective norm, and organizational culture represent the organizational norm and pressure that facilitate the use of agile methodologies. Previous research has found that rewards and norms are important factors that facilitate knowledge transfer [54,65].

Career consequences, the dimension of long-term consequences of use, refers to outcomes that have a payoff in the future, such as increasing the flexibility to change jobs or increasing the opportunities for more meaningful work [81]. When transitioning from a heavyweight process to agile methodology, some developers prefer heavyweight plan-driven processes and actively try to add formalized tasks back to an agile process because they believe those heavyweight plan-driven processes look better on a resume [18]. This suggested that career consequences will be an important incentive that motivates developers to adopt an agile methodology. With better career consequences, developers may perceive the agile methodology as more useful, and be more motivated to use it. Through active use of the methodology, developers are more likely to engage in related knowledge management activities, which in turn improve developers' acceptance of agile methodologies.

Top management support refers to the continual active and enthusiastic approval of senior executives for a proposed innovation [76]. Grover [28] found that organizations in which management are more supportive of updating technological infrastructure are more ready for innovation. In prior studies, management support has been found to be an important factor in the implementation of SDMs (e.g., [69]), and implementation success of SDMs (e.g., [32]) and agile methodologies [16]. Similar to career consequences, top management support will be an important incentive to motivate developers to use agile methodologies and engage in the associated knowledge management activities, because such behaviors will result in management's approval and possible rewards. Further, when management provide better support for the use of agile methodologies, developers are more likely to form positive perceptions toward the methodologies [38]. In sum, management support can encourage developers' knowledge management behaviors and improve their acceptance of agile methodologies.

Voluntariness is defined as "the extent to which potential adopters perceive the adoption decision to be non-mandatory" [3,31,56]. Voluntariness has been found to be a significant factor that influences the acceptance of SDMs [29,41,68]. When the use of the agile methodology is not mandatory, developers lack the incentive to use the methodology, as well as to engage in the associated knowledge management activities. As a result, developers are less likely to adopt the agile methodology, as switching to a new SDM is perceived as a radical change to the behavior processes in conducting one's work. Thus, voluntariness can discourage developers' knowledge management behaviors and make them less likely to accept agile methodologies.

Subjective norm is defined as the influence of important referents on an individual's acceptance of an SDM [30]. Previous studies found that subjective norm significantly affects intention (e.g., [84]). In particular, Venkatesh and Davis [84] found that the effect of subjective norm on intention is moderated by experience and voluntariness. In the domain of SDMs, subjective norm has been found to be an important determinant [29,30,41,68], as the emphasis on teamwork in software development creates social pressure for individuals. For agile methodologies, collaboration among developers is further emphasized, so social pressure for individuals is expected to remain influential. Prior research has suggested that norms facilitate knowledge transfer [6,65]. Thus, subjective norm can encourage developers' knowledge management behaviors and improve their acceptance of agile methodologies.

Organizational culture is defined as "a pattern of basic assumptions invented, discovered or developed by a given group as it learns to cope with its problems of external adaptation and integration that has worked well enough to be considered valid and, therefore, is to be taught to new members as the correct way to perceive, think, and feel in relation to those problems" [75]. Previous research has found that organizational culture is an important factor that can influence the acceptance and use of SDMs [41]. Further, Rus and Lindvall [73] suggested that the lack of a knowledge culture is the major obstacle to successful knowledge management in software engineering. Knowledge culture refers to "an organization that offers opportunities to create knowledge and one that encourages learning and the sharing of what is known" [52]. Without a knowledge culture, developers may be unwilling to share their knowledge because they fear that they will lose their values after the organizations capture their knowledge. Thus, the presence of an organizational culture that emphasizes knowledge sharing can encourage developers' knowledge management behaviors and improve their acceptance of agile methodologies.

4.4. Opportunity-related factors

Opportunity refers to whether individuals are provided with the opportunity to create, retain, and transfer knowledge [6]. Opportunity-related factors include teamwork, communication, shared understanding, and arduous relationship (see Table 3 for definitions). These factors refer to whether developers have good relationships among themselves (i.e., teamwork and communication) and with their customers (i.e., shared understanding, and arduous relationship) to facilitate the creation, retention, and transfer of knowledge about agile methodologies and the software product. Prior research has suggested that organizational relationships and informal networks (which can be inside or outside the organization) can help facilitate the knowledge management process [6].

Teamwork is defined as the individual's willingness to continue working with the same team as well as in other teams [82]. Walton et al. [86] suggested that the most important issue in adopting a new technology company wide, is to achieve an unusual degree of company and group unity. In previous studies on SDMs, Sultan and Chan [76] suggested that teamwork increases the likelihood of adoption of new technology, and found that adopters of object-oriented technology show higher teamwork than non-adopters. In agile development, software developers and customers are often required to work in groups and collaborate closely in order to achieve higher productivity (e.g., [13]). Good teamwork helps reduce the distance, either physically or psychologically, between people, and thus creates an environment that facilitates learning among team members. Developers who have good teamwork are more likely to learn from each other and share knowledge required for agile development. Thus, good teamwork can encourage developers' knowledge management behaviors and improve their acceptance of agile methodologies.

Communication is a process by which individuals exchange information through a common system of behavior [57]. Here, communication is defined as the extent to which software developers can communicate effectively with their peers. Prior research suggested that increased communication among peers promotes adoption and communication network links are crucial for technology adoption [72]. In systems development, frequent and accurate verbal communication among group members helps achieve higher efficiency (e.g., [16,53]) and promote knowledge transfer (e.g., [43]). Similar to the influence of teamwork, communication is an important facilitating condition that facilitates knowledge management activities among developers and the use of agile methodologies.

Shared understanding represents the extent to which the work values, norm, philosophy, problem-solving approaches, and prior work experience of a dyad are similar [26]. Shared understanding removes the barriers to understanding and acceptance between a source and a recipient, and facilitates inter-firm knowledge transfer

between a client and a consultant [46]. Shared understanding between developers and customers enables them to communicate the critical knowledge for agile development, such as product specifications and product feedback, more effectively. Further, as developers and customers are required to work closely during agile development [13], shared understanding enables developers to take advantage of the agile methodology and find it easier to use. Thus, good shared understanding between developers and customers can facilitate knowledge management activities and improve the acceptance of agile methodologies.

Arduous relationship is defined as an emotionally laborious and distant relationship between a source and a recipient [78]. A close relationship between software developers and customers will facilitate the flow and interpretation of knowledge. For instance, with a close developer–customer relationship, customers, who represent the system users, can better communicate the system requirements to software developers during the development process. Thus, contrary to shared understanding, an arduous developer–customer relationship can hinder knowledge management activities among developers and customers, and the acceptance of agile methodologies.

4.5. Agile methodology characteristics

The factors involving agile methodology characteristics include perceived usefulness, perceived ease of use, perceived compatibility, result demonstrability, and perceived maturity (see Table 3 for definitions). These factors have been found to affect user perceptions of different technology innovations, such as mobile commerce and SDMs [29,88].

Perceived usefulness refers to the degree to which an individual expects that adopting an SDM will improve the individual's job performance [29]. Prior studies on acceptance of SDMs [29,68] have found that perceived usefulness is a significant factor in predicting the acceptance of SDMs. Similar results are found in studies on other software process innovations, such as programming languages (e.g., [2]) and CASE tools (e.g., [39]). Overall, prior research suggested that the more an innovation is perceived as enabling an increase in job performance, the more likely that it will be accepted.

Perceived ease of use is defined as "the degree to which an individual believes that using a particular system would be free of physical and mental effort" [56]. It is an important factor affecting the intention to adopt an innovation in prior studies on various technologies (e.g., [20]). However, Riemenschneider et al. [68] did not find a direct effect of perceived ease of use on user acceptance in the domain of SDMs. This is possibly due to the difference in the usage contexts of IT tools and SDMs, as SDM use is more radical and mandatory, and thus reduces the relevance of how easy or how hard the behavior is to perform [68]. Moreover, since the SDM examined in prior studies (i.e., [29,68]) is based on the structured development paradigm, which is popular in the software industry, perceived ease of use may not be a crucial determinant because developers are already familiar with these traditional practices in the SDM and find them easy to use. On the other hand, the principles of agile methodologies are opposite to those of traditional SDMs, such as reduced reliance on documentation and increased acceptance of changes. Thus, perceived ease of use may have a greater effect when developers switch from traditional SDMs to agile methodologies than when developers switch to another traditional SDM. Developers are more likely to accept agile methodologies when the methodologies are easy to use.

Perceived compatibility refers to the degree to which an individual regards the practice of adopting an SDM as being consistent with his or her pre-existing software development process [29]. As the adoption of SDMs requires developers to change their existing work practices, the change in SDMs is considered to be more radical than changing tools. An SDM that is not compatible with one's work practices is unlikely to be perceived as beneficial. On the other hand, if an SDM is highly compatible with one's work practices, an individual can adopt it with least changes and benefit from its improved performance. Hence, perceived compatibility is expected to be an important factor in the domain of SDMs. Its importance is supported in prior empirical studies on traditional SDMs (e.g., [29,68]), and case studies on agile methodologies (e.g., [16,53]). Thus, developers are more likely to accept agile methodologies when the methodologies are compatible with their work practices.

Result demonstrability refers to the degree to which an innovation is perceived to be amenable to demonstration of tangible advantages [56] and is an important characteristics of any innovation [72]. While Riemenschneider et al. [68] found that result demonstrability is not a significant determinant of the acceptance of SDMs, it may be due to the fact that traditional SDMs require long development cycles, which hinder people from observing the results in a short period of time, resulting in diverse views on the result demonstrability of the SDM. On the other hand, the notable features in agile methodologies include (1) simple planning, (2) short development cycle, (3) earlier release, and (4) frequent customer feedback [36]. These features allow developers to see the results of using agile methodologies by performing frequent testing and receiving frequent customer feedback, resulting in higher result demonstrability of the agile methodology. Thus, in the case of agile methodologies, developers are more likely to accept agile methodologies when there is high result demonstrability.

Perceived maturity is a function of technology uncertainty and inexperience [49]. Cho and Kim [15] found that a technology will be more likely to be assimilated within an organization if the technology is more mature. Maturity of technology is crucial for an SDM in gaining enhancement and support from the experienced developers in the community. For example, if an SDM is mature enough, many developers will openly share their suggestions and this helps further improve the SDM, or an individual who has problems in understanding certain practices in an SDM can easily seek help from other experienced developers. On the other hand, if an SDM is immature, developers adopting it will have difficulties in getting support from the community. Thus, developers are more likely to accept agile methodologies when the methodologies are more mature.

5. Discussion

In the previous section, we have described a conceptual framework for examining the acceptance of agile methodologies. The framework suggests that acceptance of agile methodology is influenced by knowledge management outcomes (i.e., knowledge creation, knowledge retention, and knowledge transfer). In turn, knowledge management outcomes are influenced by ability-related factors (i.e., SDM self-efficacy, experience, training and external support), motivation-related factors (i.e., career consequence, top management support, voluntariness, subjective norm, and organizational culture), and opportunity-related factors (i.e., teamwork, communication, shared understanding, and arduous relationship). Further, the inclusion of agile methodology characteristics (i.e., perceived usefulness, perceived ease of use, perceived compatibility, result demonstrability, and perceived maturity) can add to the understanding of acceptance of agile methodology. This conceptual framework has both theoretical and practical contributions. In terms of theoretical contributions, the framework builds on a knowledge management perspective to provide a fresh view synthesizing various factors that can potentially influence acceptance of agile methodologies. In terms of practical contributions, the framework consolidates current knowledge on acceptance of agile methodologies which can provide guidance to organizations interested in getting their developers to use these new methodologies. We elaborate on the contributions to research and practice, and highlight some potential future research issues below.

5.1. Contributions to research

The first research contribution is in formulating a conceptual framework that incorporates existing knowledge about acceptance of SDMs and how it may apply to acceptance of agile methodologies, which have a different philosophy from the traditional SDMs. In doing so, we adopted a knowledge management perspective due to the central roles played by knowledge creation, retention, and transfer in the use of agile methodologies. The synergy between knowledge management and the knowledge aspects in agile methodologies supports the adoption of this perspective. In fact, knowledge management is recognized as an important issue in software development [73].

The second research contribution is in conducting a critical analysis of the prior literature on acceptance of SDMs and agile methodologies (which tend to be descriptive case studies). In reviewing the prior literature, we have attempted to analyze critically whether findings or factors deemed important in the domain of SDMs will also hold in the case of agile methodologies. Based on the knowledge management perspective, we have explicated Whetten's [87] three elements of theory development: the "what," "how," and "why" effects of various potential factors on the acceptance of agile methodologies.

The third research contribution is in identifying a wider set of factors that can influence acceptance of agile methodologies. Prior research on the acceptance of SDMs has treated SDMs as technology innovations and made heavy use of the technology adoption perspective. Our review has identified other important characteristics (i.e., ability-related factors, motivation-related factors, and opportunity-related factors), besides technological characteristics, that can potentially influence the acceptance of agile methodologies. By incorporating a more diverse group of characteristics into the framework, we provide a more complete understanding of the acceptance of agile methodologies.

The fourth contribution is the conceptual framework can be a foundation for future empirical research, which is currently inadequate in the domain of agile methodologies [22]. Although parts of our framework have been validated by prior research on traditional SDMs, no empirical investigation has been undertaken in the domain of agile methodologies. Due to the significant differences in philosophy between traditional SDMs and agile methodologies [33], which lead to numerous challenges in transitioning to agile methodologies [59], further validation in the domain of agile methodologies is very much needed.

5.2. Future research

There are many avenues for future research into the acceptance of agile methodologies. We will highlight two fruitful avenues. The first avenue is to empirically validate our conceptual framework. Future theoretical development may be advanced by empirically validating the crucial factors that have dominant effect on the acceptance of agile methodologies. Prior studies on the acceptance of technology have suggested that characteristics of the technology are significant determinants of the acceptance of various technologies or IT tools (e.g., [3,14]). However, it is unclear whether technology characteristics will still dominate in the agile methodologies context, because no existing study has empirically examined the various characteristics in our proposed conceptual framework. Moreover, Sultan and Chan [76] noted that characteristics of technology may not be crucial when the subjects are experienced programmers. Empirical testing of our conceptual framework will be very valuable in understanding the weight that organizations should place on different characteristics in promoting developers' acceptance of agile methodologies.

A second avenue of research is to tease out the effects of different characteristics on the three knowledge management outcomes respectively. For example, communication may have a stronger influence on knowledge transfer than on knowledge creation and retention, as communication in software engineering is often related to knowledge transfer [73]. Moreover, organizational culture is a broad concept that has many dimensions. In this paper, we focus only on the knowledge aspect—i.e., the knowledge culture in organizations. As organizational culture can be operationalized as multiple dimensions [41], they may influence the process of accepting agile methodologies differently. Future research can examine different operationalizations of organizational culture, as well as cultural differences between individual developers (e.g., [44]).

5.3. Contributions to practice

There are a number of contributions to practice based on our conceptual framework. First, the conceptual framework highlights the need for organizations to consider multiple perspectives in deploying agile methodologies among systems developers. Besides using the dominant technology adoption perspective (which emphasizes characteristics of the technology), organizations can also adopt a knowledge management perspective in parallel to have a more comprehensive understanding of how to develop acceptance of agile methodologies. A knowledge management perspective is valuable as there are related knowledge issues in the use of agile methodologies.

Second, the conceptual framework lists a number of facilitating factors that can lead to acceptance of agile methodologies. While technology characteristics are an important group of factors to consider, organizations should not neglect other characteristics (such as ability-related factors, motivation-related factors, and opportunity-related factors), as all of them can potentially influence acceptance of agile methodologies. This expanded list of facilitating factors can act as levers which organizations can use in devising strategies to promote acceptance of agile methodologies among their systems developers. Besides educating systems developers on the positive characteristics of agile methodologies (e.g., its usefulness, ease of use, compatibility, results demonstrability, and maturity), organizations can implement initiatives (such as providing adequate training, external support, etc.) to improve the ability of their systems developers in using agile methodologies, Organizations can also influence systems developers' motivations by providing incentives (e.g., emphasizes positive career consequences and availability of top management support) and socializing employees on organizational norms (e.g., voluntariness, subjective norms, and organizational culture) to encourage acceptance of new technologies, such as agile methodologies. Finally, organizations may create opportunities to develop a knowledge culture (e.g., teamwork, communication, shared understanding, interpersonal relationship) by establishing small group meeting rooms, and rewarding those who pursue learning and who teach others what they know [52]. This can facilitate the knowledge management practices which are crucial to using agile methodologies effectively.

Third, our conceptual framework helps explain why agile methodologies may not be suitable for all projects. The various characteristics included in the framework capture certain important underlying assumptions about agile methodologies. For example, in using agile methodologies, there must exist good communication among the team members, as the transfer of knowledge (e.g., product specification) relies heavily on human-to-human communication. Developers must also have a shared understanding with their customers, as extensive customer interaction is needed throughout the development process. If some of these assumptions are not satisfied in a particular project, then the agile methodologies may not be appropriate for that project [77].

6. Conclusion

In summary, this article has developed a conceptual framework to understand acceptance of agile methodologies. Our study contributes to the systems development literature in several ways. First, we have

conducted a critical analysis of the prior literature on acceptance of traditional SDMs and the newer agile methodologies. We have synthesized findings from the prior literature and analyzed the applicability of the identified factors in the domain of agile methodologies. Second, we introduce knowledge management as a fresh perspective to examine the acceptance of agile methodologies. This conceptual framework can be used to understand how various characteristics affect developers' decision to accept agile methodologies, and is amenable to empirical testing. Finally, we provide some guidelines to effect acceptance of agile methodologies.

Acknowledgement

This research is partially funded by grant SBI08/09. BM07 from the Hong Kong University of Science and Technology.

References

[1] P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta, Agile Software Development Methods: Review and Analysis, VTT Publications, 2002.
[2] R. Agarwal, J. Prasad, A field study of the adoption of software process innovations by information systems professionals, IEEE Transactions on Engineering Management 17 (3) (2000) 295–308.
[3] R. Agarwal, J. Prasad, The role of innovation characteristics and perceived voluntariness in the acceptance of information technologies, Decision Sciences 28 (3) (1997) 557–582.
[4] R. Agarwal, V. Sambamurthy, R.M. Stair, Research report: the evolving relationship between general and specific computer self-efficacy—an empirical assessment, Information Systems Research 11 (4) (2000) 418–430.
[5] S.W. Ambler, Results from Scott Ambler's March 2006 'Agile Adoption Rate Survey', 2006 http://www.ambysoft.com/surveys/agileMarch2006.html, accessed October 15, 2007.
[6] L. Argote, B. McEvily, R. Reagans, Managing knowledge in organizations: an integrative framework and review of emerging themes, Management Science 49 (4) (2003) 571–582.
[7] P. Behrens, Agile Project Management (APM) Tooling Survey Results, , 2006 http://www.trailridgeconsulting.com/files/2006AgileToolingSurveyResults.pdf, accessed October 15, 2007.
[8] B. Boehm, Get ready for agile methods, with care, IEEE Computer 35 (1) (2002) 64–69.
[9] B. Boehm, R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed, Addison-Wesley, Boston, MA, 2004.
[10] R.E. Brooks, Studying programmer behavior experimentally: the problems of proper methodology, Communications of the ACM 23 (4) (1980) 207–213.
[11] M.E. Burkhardt, D.J. Brass, Changing patterns or patterns of change: the effects of change in technology on social network structure and power, Administrative Science Quarterly 35 (1990) 104–127.
[12] A. Cabrera, W.C. Collins, J.F. Salgado, Determinants of individual engagement in knowledge sharing, International Journal of Human Resource Management 17 (2) (2006) 245–264.
[13] M. Ceschi, A. Sillitti, G. Succi, S.D. Panfilis, Project management in plan-based and agile companies, IEEE Software 22 (3) (2005) 21–27.
[14] P.Y.K. Chau, An empirical assessment of a modified technology acceptance model, Journal of Management Information Systems 13 (2) (1996) 185–204.
[15] I. Cho, Y.G. Kim, Critical factors for assimilation of object-oriented programming languages, Journal of Management Information Systems 18 (3) (2001) 125–156.
[16] A. Cockburn, J. Highsmith, Agile software development: the people factor, IEEE Computer 34 (11) (2001) 131–133.
[17] W.M. Cohen, D. Levinthal, Absorptive capacity: a new perspective on learning and innovation, Administrative Science Quarterly 35 (1990) 128–152.
[18] M. Cohn, D. Ford, Introducing an agile process to an organization, IEEE Computer 36 (6) (2003) 74–78.
[19] D.R. Compeau, C.A. Higgins, Computer self-efficacy: development of a measure and initial test, MIS Quarterly 19 (2) (1995) 189–211.
[20] F.D. Davis, Perceived usefulness, perceived ease of use, and user acceptance of information technology, MIS Quarterly 13 (3) (1989) 319–340.
[21] J. Drobka, D. Noftz, R. Raghu, Piloting XP on four mission-critical projects, IEEE Software 21 (6) (2004) 70–75.
[22] J. Erickson, K. Lyytinen, K. Siau, Agile modeling, agile software development, and extreme programming: the state of research, Journal of Database Management 16 (4) (2005) 88–100.
[23] R.G. Fichman, C.F. Kemerer, Adoption of software engineering process innovations: the case of object orientation, Sloan Management Review 34 (2) (1993) 7–22.
[24] R.G. Fichman, C.F. Kemerer, The assimilation of software process innovations: an organizational learning perspective, Management Science 43 (10) (1997) 1345–1363.
[25] B. Fitzgerald, Formalised systems development methodologies: a critical perspective, Information Systems Journal 6 (1) (1996) 3–23.
[26] D. Gerwin, L. Moffat, Withdrawal of team autonomy during concurrent engineering, Management Science 43 (9) (1997) 1275–1287.
[27] R. Glass, A snapshot of systems development practice, IEEE Software 16 (3) (1999) 110–111.
[28] V. Grover, An empirically derived model for the adoption of customer-based interorganizational systems, Decision Sciences 23 (4) (1993) 603–640.
[29] B. Hardgrave, F.D. Davis, C. Riemenschneider, Investigating determinants of software developers' intentions to follow methodologies, Journal of Management Information Systems 20 (1) (2003) 123–151.
[30] B. Hardgrave, R.A. Johnson, Toward an information systems development acceptance model: the case of object-oriented systems development, IEEE Transactions on Engineering Management 50 (3) (2003) 322–336.
[31] J. Hartwick, H. Barki, Explaining the role of user participation in information system use, Management Science 40 (4) (1994) 440–465.
[32] S.H. Higgins, P.T. Hogan, Internal diffusion of high technology industrial innovations: an empirical study, Journal of Business & Industrial Marketing 14 (1) (1999) 61–75.
[33] J. Highsmith, Agile Software Development Ecosystems, Addison-Wesley, 2002.
[34] J. Highsmith, A. Cockburn, Agile software development: the business of innovation, IEEE Computer 34 (9) (2001) 120–122.
[35] T. Hill, N.D. Smith, M.F. Mann, Role of efficacy expectations in predicting the decision to use advanced technologies: the case of computers, Journal of Applied Psychology 72 (1987) 307–313.
[36] M. Huo, J. Verner, L. Zhu, M.A. Babar, Software quality and agile methods, Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04), 2004, pp. 520–525.
[37] M. Igbaria, J. Iivari, The effects of self-efficacy on computer usage, Omega 23 (6) (1996) 587–605.
[38] M. Igbaria, N. Zinatelli, P. Cragg, A. Cavaye, Personal computing acceptance factors in small firms: a structural equation model, MIS Quarterly 21 (3) (1997) 279–305.
[39] J. Iivari, Why are CASE tools not used? Communications of the ACM 39 (10) (1996) 94–103.
[40] J. Iivari, R. Hirschheim, H.K. Klein, A dynamic framework for classifying information systems development methodologies and approaches, Journal of Management Information Systems 17 (3) (2000) 179–218.
[41] J. Iivari, M. Huisman, The relationship between organizational culture and the deployment of systems development methodologies, MIS Quarterly 31 (1) (2007) 35–58.
[42] R.A. Johnson, Applying the technology acceptance model to a systems development methodology, Proceedings of Americas Conference on Information Systems, 1999, pp. 572–573.
[43] K.D. Joshi, S. Sarker, S. Sarker, Knowledge transfer within information systems development teams: examining the role of knowledge source attributes, Decision Support Systems 43 (2) (2007) 322–335.
[44] A. Kankanhalli, B.C.Y. Tan, K.K. Wei, M.C. Holmes, Cross-cultural differences and information systems developer values, Decision Support Systems 38 (2) (2004) 183–195.
[45] M. Khalifa, J.M. Verner, Drivers for software development method usage, IEEE Transactions on Engineering Management 47 (3) (2000) 360–369.
[46] D.G. Ko, L.J. Kirsch, W.R. King, Antecedents of knowledge transfer from consultants to clients in enterprise system implementations, MIS Quarterly 29 (1) (2005) 59–85.
[47] A.S. Koch, Agile Software Development: Evaluating the Methods for Your Organization, Artech House, Boston, 2005.
[48] K.A. Kozar, Adopting systems development methods: an exploratory study, Journal of Management Information Systems 5 (4) (1989) 73–86.
[49] G.H. Lee, Y.G. Kim, Implementing a client/server system in Korean organizations: interrelated IT innovation perspective, IEEE Transactions on Engineering Management 45 (3) (1998) 287–295.
[50] L. Lindstrom, R. Jeffries, Extreme programming and agile software development methodologies, Information Systems Management 21 (3) (2004) 41–60.
[51] W. McGehee, P.W. Thayer, Training in Business and Industry, Wiley, New York, 1961.
[52] C. McInerney, Knowledge management and the dynamic nature of knowledge, Journal of the American Society for Information Science and Technology 53 (12) (2002) 1009–1018.
[53] J. McManus, Team agility, Computer Bulletin 45 (5) (2003) 26–27.
[54] T. Menon, J. Pfeffer, Valuing internal versus external knowledge, Management Science 49 (4) (2003) 497–513.
[55] K. Mohan, B. Ramesh, Traceability-based knowledge integration in group decision and negotiation activities, Decision Support Systems 43 (3) (2007) 968–989.
[56] G.C. Moore, I. Benbasat, Predicting user intentions: comparing the technology acceptance model with the theory of planned behavior, Information Systems Research 2 (3) (1991) 192–222.
[57] J.G. Myers, Diffusion and Personal Influence: Determinants of Adoption Rates, Opinion Leaders' Characteristics, and Venturesomeness, Working Paper, vol. 88, University of California, Berkeley, CA, 1972.
[58] J. Nadler, L. Thompson, L. Van Boven, Learning negotiation skills: four models of knowledge creation and transfer, Management Science 49 (4) (2003) 529–540.
[59] S. Nerur, R. Mahapatra, G. Mangalaraj, Challenges of migrating to agile methodologies, Communications of the ACM 48 (5) (2005) 73–78.
[60] S.K. Parker, From passive to proactive motivation: the importance of flexible role orientations and role breadth self-efficacy, Applied Psychology: An International Review 49 (3) (2000) 447–469.
[61] A. Parrish, R. Smith, D. Hale, J. Hale, A field study of developer pairs: productivity impacts and implications, IEEE Software 21 (5) (2004) 76–79.
[62] S.L. Pfleeger, Understanding and improving technology transfer in software engineering, Journal of Systems and Software 47 (2–3) (1999) 111–124.

[63] S. Raghunathan, A structured modeling based methodology to design decision support systems, Decision Support Systems 17 (4) (1996) 299–312.
[64] T. Ravichandran, A. Rai, Total quality management in information systems development, Journal of Management Information Systems 16 (3) (1999) 119–155.
[65] R. Reagans, B. McEvily, Network structure and knowledge transfer: the transfer problem revisited, Working Paper, Columbia University, New York, 2003.
[66] D.J. Reifer, How good are agile methods? IEEE Software 19 (4) (2002) 16–18.
[67] C.K. Riemenschneider, B.C. Hardgrave, Explaining software development tool use with the technology acceptance model, Journal of Computer Information Systems 41 (4) (2001) 1–8.
[68] C.K. Riemenschneider, B.C. Hardgrave, F.D. Davis, Explaining software developer acceptance of methodologies: a comparison of five theoretical models, IEEE Transactions on Software Engineering 28 (12) (2002) 1135–1145.
[69] T.L. Roberts, M.L. Gibson, K.T. Fields, R.K. Rainer, Factors that impact implementing a system development methodology, IEEE Transactions on Software Engineering 24 (8) (1998) 640–649.
[70] T.L. Roberts, C.T. Hughes, Obstacles to implementing a system development methodology, Journal of Systems Management 47 (2) (1996) 36–40.
[71] T.L. Roberts, W. Leigh, R.L. Purvis, M.J. Parzinger, Utilizing knowledge links in the implementation of system development methodologies, Information and Software Technology 43 (2001) 635–640.
[72] E.M. Rogers, Diffusion of Innovations, 4th ed.Free Press, New York, 1995.
[73] I. Rus, M. Lindvall, Knowledge management in software engineering, IEEE Software 19 (3) (2002) 26–38.
[74] B. Schatz, I. Abdelshafi, Primavera gets agile: a successful transition to agile development, IEEE Software 22 (3) (2005) 36–42.
[75] E.H. Schein, Organizational culture, American Psychologist 45 (2) (1990) 109–119.
[76] F. Sultan, L. Chan, The adoption of new technology: the case of object-oriented computing in software companies, IEEE Transactions on Engineering Management 47 (1) (2000) 106–126.
[77] N.C. Surendra, Using an ethnographic process to conduct requirements analysis for agile systems development, Information Technology and Management 9 (1) (2007) 55–69.
[78] G. Szulanski, Exploring internal stickiness: impediments to the transfer of best practice within the firm, Strategic Management Journal 17 (1996) 27–43.
[79] G.F. Templeton, T.A. Byrd, Determinants of the relative advantage of a structured SDM during the adoption stage of implementation, Information Technology and Management 4 (2003) 409–428.
[80] L. Thompson, D. Gentner, J. Lowenstein, Avoiding missed opportunities in managerial life: analogical training more powerful than individual case training, Organizational Behavior and Human Decision Processes 82 (2000) 60–75.
[81] R. Thompson, C. Higgins, J. Howell, Personal computing: toward a conceptual model of utilization, MIS Quarterly 15 (1) (1991) 125–143.
[82] B.C.R. Ulloa, S.G. Adams, Attitude toward teamwork and effective teaming, Team Performance Management 10 (7/8) (2004) 145–152.
[83] V. Venkatesh, Creation of favorable user perceptions: exploring the role of intrinsic motivation, MIS Quarterly 23 (2) (1999) 239–260.
[84] V. Venkatesh, F.D. Davis, A theoretical extension of the technology acceptance model: four longitudinal field studies, Management Science 46 (2) (2000) 186–204.
[85] I. Vessey, R.L. Glass, Strong vs. weak approaches to systems development, Communications of the ACM 41 (4) (1998) 99–102.
[86] R.E. Walton, J.M. Dutton, T.P. Cafferty, Organizational context and interdepartmental conflict, Administrative Science Quarterly 14 (1969) 522–542.
[87] D.A. Whetten, What constitutes a theoretical contribution? Academy of Management Review 14 (4) (1989) 490–495.
[88] J.H. Wu, S.C. Wang, What drives mobile commerce? An empirical evaluation of the revised technology acceptance model, Information & Management 42 (2005) 719–729.
[89] J.L. Wynekoop, N.L. Russo, Systems development methodologies: unanswered questions, Journal of Information Technology 10 (1995) 65–73.
[90] H. Zhuge, Component-based workflow systems development, Decision Support Systems 35 (4) (2003) 517–536.

Frank K.Y. Chan is a doctoral student in Information Systems at the School of Business and Management, Hong Kong University of Science and Technology. He earned his bachelor degree in business administration (1st class honors) from the Hong Kong University of Science and Technology. His research interests include IT adoption, electronic government, and systems development.

James Y.L. Thong is Professor of Information Systems at the HKUST Business School, Hong Kong University of Science and Technology. His research on technology adoption, human–computer interaction, computer ethics, and IT in small business has appeared in *Information Systems Research*, *Journal of Management Information Systems*, *Communications of the ACM*, *Decision Support Systems*, *European Journal of Information Systems*, *European Journal of Operational Research*, *International Journal of Human–Computer Studies*, *Journal of the American Society for Information Science and Technology*, *Journal of Information Technology*, and *Telecommunications Policy*, among others. He has served as an Associate Editor for *MIS Quarterly* and Guest Editor for *Decision Support Systems*, *International Journal of Electronic Commerce*, *Journal of Organizational Computing and Electronic Commerce*, and *Journal of Global Information Management*.