2007

An Integrated Framework of Individual Acceptance of Agile Methodologies

K.Y. Chan
Hong Kong University of Science and Technology

James Y.L Thong
Hong Kong University of Science and Technology

Follow this and additional works at: http://aisel.aisnet.org/pacis2007

Recommended Citation
http://aisel.aisnet.org/pacis2007/154

This material is brought to you by the Pacific Asia Conference on Information Systems (PACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in PACIS 2007 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
An Integrated Framework of Individual Acceptance of Agile Methodologies

K.Y. Chan  
Hong Kong University of Science and Technology

James Y.L. Thong  
Hong Kong University of Science and Technology

Abstract

Although systems development methodologies (SDM) are believed to help improve the software development process, the deployment of SDMs often encounters developers’ resistance. Agile methodologies, the latest batch of SDMs that are better in dealing with volatile business requirements, are likely to face the same problems since they require developers to drastically change their work habits and acquire new skills. Therefore, it is essential to identify the factors that could potentially influence individual developer’s acceptance of agile methodologies. This paper reviews the literature on the acceptance of SDMs and agile methodologies, which serves to assess current understanding in the acceptance of SDMs and agile methodologies. From this review, we identify a set of potential factors and develop a framework to guide future research in individual acceptance of agile methodologies.

Keywords: systems development methodologies, agile methodologies, adoption

Introduction

Systems development methodology (SDM), defined as a documented collection of policies, processes, and procedures used by a development team or an organization to practice software engineering, can help improve the software development process in terms of productivity and quality (Iivari et al., 2000). Examples of traditional SDMs include Structured Systems Analysis and Design Methodology and Object-oriented Systems Development (OOSD). However, only about half of all organizations actually follow an SDM (Glass, 1999). These organizations often encounter resistance from individual software developers when the organizations attempt to deploy an SDM (Kozar, 1989). As a result, the expenditure of time and effort in implementing an SDM makes it one of the most serious areas of concern in Information Systems (IS) research (Roberts et al., 1998).

SDMs are constantly evolving to keep up with changing technologies and satisfy new demands from users. A new batch of SDMs called agile methodologies claim to be better in dealing with such a dynamic business environment (Koch, 2005). Agile methodologies such as Extreme Programming (XP) and Scrum, have received praise from practitioners because of their abilities to deal with volatile requirements. For those organizations that have employed traditional SDMs for years, migrating to agile methodologies will pose challenges in many aspects, such as management style, systems development process, collaboration of work groups, and compatibility of technologies (Nerur et al., 2005), in addition to resistance from individual software developers.

Agile development emphasizes the people factor (Cockburn and Highsmith, 2001) – both software developers and users play an important role in agile development. In traditional
system development, users participate mainly in specification development, but their participation is minimal in other activities (Nerur et al., 2005). On the other hand, in agile development, customers (representing system users) work in small teams with developers as active team members. For example, customers and developers jointly decide the system features to be implemented in each development cycle. Such a drastic change of user’s role suggests that the successful adoption of agile methodology is concerned with not only software developers and organizations, but also customers, who are expected to be collaborative, representative, authorized, committed, and knowledgeable (Boehm and Turner, 2004). Therefore, the relationship with customers is likely to be an important consideration when software developers decide whether to adopt an agile methodology, because it is not easy to find those “qualified” customers, especially for complex systems (Nerur et al., 2005).

The goal of this paper is to identify factors that are critical to the individual developer’s acceptance of agile methodologies. In this paper, we first review the empirical studies on the acceptance of traditional SDMs. Next, we review studies on agile methodologies to identify potential factors that may contribute to the acceptance of agile methodologies. We then incorporate the findings from the reviews into a research framework that will provide a basis for future research into the acceptance of agile methodologies. Finally, we discuss the implications for research and practice.

**Background**

**Barriers to the adoption 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 (Fitzgerald, 1996).

Despite the potential advantages of using SDMs, practitioners have been slow in adopting SDMs. Fitzgerald (1998) 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 adoption rate are: 1) developers simply ignore the newly introduced SDMs, 2) SDMs treat the process of systems development as an orderly rational process while it is not, 3) SDMs are assumed to be universally applicable, but they should be adjusted across different development situations (Fitzgerald, 1998). On the whole, the actual adoption of SDMs among practitioners has remained a perennial question that attracts researchers’ interest.

**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 has outlined the principles of agile systems development (Lindstrom and Jeffries, 2004). 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. An overview of these leading agile methods can be found in Highsmith (2002).

Highsmith (2002, P. 32) suggested that differences between traditional SDMs and agile methodologies rested 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 a full knowledge in system requirements at the beginning (Highsmith, 2002). Therefore, 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 (Highsmith, 2002). Second, traditional SDMs assume customers are short-sighted, and thus developers have to build in extra functionalities to suit the arising needs of customers (Highsmith, 2002). Highsmith (2002) suggested that this assumption often led to over-designed systems, which attempted to solve future problems. On the other hand, agile methodologies emphasize simplicity — the art of maximizing the work not done (Lindstrom and Jeffries, 2004). Moreover, 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 (Nerur et al, 2005).

These significant differences bring a number of challenges to successful transition to agile methodologies. Nerur et al. (2005) suggested that migrating to agile methodologies involved issues related to various aspects, including management, people, process and technology. The complications in migrating to agile methodologies make it unlikely that previous findings on the adoption of traditional SDMs would be readily applicable to the adoption of 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. Furthermore, previous studies on the adoption of SDMs (e.g., Hardgrave and Johnson, 2003; Hardgrave et al., 2003; Riemenschneider et al., 2002) focusing on developers’ beliefs about an SDM (e.g., perceived usefulness and perceived ease of use) failed to take into account the importance of people-related issues (e.g., individual competence) and management-related issues (e.g., management style).

**Prior Literature**

**Identification of the relevant literature base**

Our study starts with assessing the current state of knowledge with respect to understanding the acceptance of traditional SDMs, such as Structured Systems Analysis and Design Methodology, Waterfall Model, and Object-oriented Systems Development (OOSD). In particular, OOSD is occasionally regarded as a programming language (e.g., C language) instead of an SDM. However, Fichman and Kemerer (1993, p. 19) 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. Therefore, OOSD is regarded as an SDM and studies on OOSD are 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 systems development process and are, thus, much less complex (Vessey and Glass, 1998), however, adoption of a systems development process represents a much more radical change than adopting tools or techniques (Roberts et al., 1998). As a result, factors that are important to the adoption of tools or techniques may not be applicable to the domain of SDMs. As found in our review, the prior literature on the adoption of SDMs consists of two main streams,
focusing on 1) individual developer’s adoption of SDMs and 2) organizational adoption and implementation of SDMs.

The second part of our review focused on studies on agile methodologies. Although there were a large number of studies on agile methodologies, these studies focused mainly on the practical usage of agile methodologies (e.g., Boehm, 2002) and the costs and benefits which the agile methodologies brought (e.g., Parrish et al., 2004), whereas no empirical study attempted to examine factors affecting the adoption of agile methodologies. In a recent research review on agile methodology, Erickson et al. (2005) reported similar findings that previous research mainly consisted of case studies, comparative analysis and experience reports, whereas very few empirical studies have been conducted. In total, we identified 7 articles that empirically examined individual acceptance of SDMs (Hardgrave and Johnson, 2003; Hardgrave et al., 2003; Johnson, 1999; Khalifa and Verner, 2000; Riemensneider et al., 2002; Sultan and Chan, 2000; Templeton and Byrd, 2003), 3 articles that empirically examined organizational acceptance of SDMs (Cho and Kim, 2001; Fichman and Kemerer, 1997; Higgins and Hogan, 1999), 3 studies that aimed to identify important factors in the implementation of SDMs (Roberts and Hughes, 1996; Roberts et al., 1998; Roberts et al., 2001), and 8 case studies that discussed the adoption of agile methodologies (Ceschi et al., 2005; Cockburn and Highsmith, 2001; Cohn and Ford, 2003; Drobka et al., 2004; Highsmith and Cockburn, 2001; McManus, 2003; Nerur et al., 2005; Schatz and Abdelshafi, 2005). The lack of empirical efforts in this area justifies the need for developing a research framework for the acceptance of agile methodologies, which forms a basis for future empirical research.

Acceptance of systems development methodologies

Existing theories for examining the acceptance of Information Technology (IT) tools (e.g., Innovation Diffusion Theory (Rogers, 1995); Technology Acceptance Model (TAM; Davis, 1989)) have been widely used for examining individual intentions to adopt IT innovations, such as the World Wide Web (e.g. Agarwal and Prasad, 1997) and spreadsheets (e.g. Chau, 1996). However, the applicability of these models in the domain of acceptance of SDMs has received little researchers’ attention.

Johnson (1999) 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. Riemensneider et al. (2002) 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. These models were found to have moderate predictive power ($R^2 \geq .50$). Among eleven 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 similar study, Hardgrave et al. (2003) investigated the determinants of individual developer’s intentions to follow methodologies, based on two major 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 (2003) reported similar findings that perceived ease of use and perceived compatibility were significant determinants of relative advantage, which is regarded as the same as perceived usefulness in TAM (Moore and Benbasat, 1991).
On the other hand, some previous studies focused on characteristics of individual developers (e.g., experience) and organizations (e.g., management support), in addition to the characteristics of SDMs. Sultan and Chan (2000) reported that characteristics of the technology, i.e., relative advantage, perceived compatibility and complexity, were not significant in differentiating adopters and non-adopters of OO technology. The results contradict the findings in other studies (e.g., Hardgrave et al., 2003), 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 decision to adopt it because of other factors such as characteristics of individual developers and organizations (Sultan and Chan, 2000).

Another branch of research focuses on organizational adoption and implementation of SDMs. Studies on the assimilation of OO technology (e.g., Cho and Kim, 2001; Fichman and Kemerer, 1997; Roberts et al., 1998) suggested that learning-related factors, such as training and external support, were positively related to organizational assimilation of OO technology. These studies, like studies on individual acceptance of SDMs, suggested that management support was crucial to the successful assimilation of SDMs. However, these studies did not take characteristics of developers or characteristics of SDMs into consideration, or found these characteristics to be insignificant. In Cho and Kim’s study (2001), characteristics of developers (i.e., average years of professional experience) and characteristics of SDMs (i.e., relative advantage and complexity) were found to be insignificant in affecting the assimilation of OO technology, but interestingly, perceived maturity of technology was found to be significant. On the whole, these studies identified a number of organizational factors (e.g., management support and training) that were also found to be significant determinants of individual acceptance of SDMs, and suggested some additional characteristics of SDMs (e.g., maturity of technology) that may contribute to individual acceptance of SDMs.

Acceptance of agile methodologies
Most of these previous 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 SDMs, studies on agile methodologies suggested that individual characteristics (i.e., ability of developers) were important in deploying agile methodologies (e.g. Ceschi et al., 2005; Cohn and Ford, 2003) and using agile methodologies effectively (e.g. Cockburn and Highsmith, 2001). This may be due to the fact that agile methodologies emphasize the people factor (Cockburn and Highsmith, 2001).

The characteristics of agile methodologies, such as compatibility, were also found to be important factors affecting the acceptance of agile methodologies (e.g. Cockburn and Highsmith, 2001; McManus, 2003; Schatz and Abdelshafi, 2005). Moreover, Nerur et al. (2005) pointed out some important organizational characteristics, such as organizational culture, teamwork, and training, consistent with the findings by Sultan and Chan (2000).

One distinction between studies on 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 often examine factors associated with users, such as customer relationships (e.g., Ceschi et al., 2005; Nerur et al., 2005). This may be 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 (Nerur et al., 2005).
Limitations of existing research: The need for an integrated framework

In summary, previous 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., Riemenschneider et al., 2002). 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 characteristics of SDMs in some cases (e.g., Sultan and Chan, 2000). Third, studies on organizational acceptance of SDMs (e.g., Cho and Kim, 2001) provide additional characteristics of SDMs (e.g., maturity of technology) that may influence the adoption of agile methodologies. Finally, 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 (e.g., Nerur et al., 2005). This suggests that factors associated with customers would be crucial to the adoption of agile methodologies.

While previous studies have empirically examined some of the above factors, it is still a fragmented piece of work, with no study considering the combined influences of these factors. Below, we propose a comprehensive research framework that incorporates relevant factors which can affect the individual developer’s acceptance of agile methodologies.

Research Framework

Based on previous empirical studies on individual/organizational acceptance of SDMs and case studies on agile methodologies, a number of potential factors are identified for inclusion in the research framework. Table 1 presents an overview of potential determinants of the individual acceptance of agile methodologies that were identified from previous studies on SDMs and agile methodologies. These determinants are categorized into 1) individual characteristics—factors associated with software developers; 2) organizational characteristics—factors associated with the management and the organization; 3) agile methodology characteristics—factors associated with agile methodology; and 4) customer relationships—factors associated with the developer-customer relationship. In particular, since customers do not play an active role in traditional development process, previous research in the acceptance of SDMs has not studied factors related to the developer-customer relationship. To fill this gap, we drawn from the literature of knowledge management to identify constructs to capture the characteristics of developer-customer relationships. The conceptual framework is illustrated in Figure 1. The signs in parentheses indicate the proposed association between the determinants and software developers’ intention to adopt agile methodology.
**Individual characteristics**

SDM self-efficacy is defined as a judgment of one’s capability to use an SDM, (Compeau et al., 1995). Previous studies have suggested the importance of self-efficacy in computing and high technology domains (Agarwal et al., 2000). For example, self-efficacy has been found to affect computer use (e.g., Igbaria and Iivari, 1995), early adoption of computer system (e.g., Burkhardt and Brass, 1990), and behavioral intention (e.g., Hill et al., 1987). In the domain of agile methodologies, individual ability of software developers is often stressed, because if the people on the project are good enough, they can use almost any process to accomplish their tasks (Cockburn and Highsmith, 2001). Thus, software developers with high SDM self-efficacy are more likely to be able to use the agile methodology and to adopt it.

Experience is defined as encounters that one undergoes or lives through (Sultan and Chan, 2000). Hill et al. (1987) suggested that experienced individuals had encountered more decisions and perceived an increase in their ability to make innovation decisions. Thus, experience of individuals should have a positive impact on acceptance of new technologies. 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 (Brooks, 1980). Furthermore, agile development tends to rely on developers’ knowledge of different methods, such as XP, Crystal and Scrum (McManus, 2003). With more experience in using various SDMs, a developer would have a greater ability to recognize the advantages of the agile methodology, resulting in more positive beliefs towards the agile methodology. Therefore, software developers with more experience are more likely to adopt agile methodology.

Teamwork is defined as the individual’s willingness to continue working with the same team as well as in other teams (Ulloa and Adams, 2004). Walton et al. (1969) 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 (2000) suggested that teamwork increased the likelihood of adoption of new technology, and found that adopters of object-oriented technology showed 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., Ceschi et al., 2005). On the whole, good teamwork is expected to encourage the adoption of agile methodology by providing a more conducive environment for adoption of new technologies, as well as a facilitating atmosphere for agile development. Therefore, software developers are more likely to adopt agile methodology when they have good teamwork.

**Communication** is a process by which individuals exchange information through a common system of behavior (Myers, 1972). 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 (Rogers, 1983). In agile development, frequent and accurate verbal communication among group members helps achieve higher efficiency (e.g., McManus, 2003). Therefore, software developers are more likely to adopt agile methodology when they have good communication within their work groups.

<table>
<thead>
<tr>
<th>Table 1: Potential Determinants of Individual Adoption of Agile Methodologies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Category</strong></td>
</tr>
<tr>
<td>Individual characteristics</td>
</tr>
<tr>
<td>Experience</td>
</tr>
<tr>
<td>Teamwork</td>
</tr>
<tr>
<td>Career consequences</td>
</tr>
<tr>
<td>Organizational characteristics</td>
</tr>
<tr>
<td>External support</td>
</tr>
<tr>
<td>Voluntariness</td>
</tr>
<tr>
<td>Subjective norm</td>
</tr>
</tbody>
</table>
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” (McGehee and Thayer, 1961, p. 10).

Cho and Kim (2001); Riemenschneider and Hardgrave (2001); Roberts and Hughes (1996)

Perceived usefulness refers to the degree to which an individual expects that adopting an SDM will improve his or her individual job performance (Hardgrave et al., 2003).

Hardgrave et al. (2003); Riemenschneider et al. (2002)

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 (Moore and Benbasat, 1991).

Hardgrave et al. (2003); Riemenschneider et al. (2002)

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 (Hardgrave et al., 2003).

Cho and Kim (2001); Hardgrave et al. (2003); Riemenschneider et al. (2002)

Result demonstrability refers to the degree to which an innovation is perceived to be amenable to demonstration of tangible advantages (Moore and Benbasat, 1991).

Kiemenschneider et al. (2002)

Maturity of technology refers to the degree of uncertainty and inexperience (Lee and Kim, 1998).

Cho and Kim (2001)

Shared understanding represents the extent to which the work values, norm, philosophy, problem-solving approaches, and prior work experience of a dyad are similar (Gerwin and Moffat, 1997).

N/A

Arduous relationship is defined as an emotionally labious and distant relationship between a source and a recipient (Szulanski, 1996).

N/A

Ceschi et al. (2005); Nerur et al. (2005)

**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 (Thompson et al., 1991). When transitioning from a heavyweight process to agile methodology, some developers preferred heavyweight plan-driven processes and actively tried to add formalized tasks back to an agile process because they believed those heavyweight plan-driven processes looked better on a resume (Cohn and Ford, 2003). Therefore, it is expected that career consequences would be an important consideration when developers decide whether to adopt an agile methodology.

**Organizational characteristics**

**Top management support** is the continual active and enthusiastic approval of senior executives for a proposed innovation (Sultan and Chan, 2000). Grover (1993) found that organizations in which management were more supportive of updating technological infrastructure were more ready for innovation. In previous studies, management support was found to be an important factor in the implementation of SDMs (e.g., Roberts et al., 1998), and implementation success of SDMs (e.g., Higgins and Hogan, 1999) and agile methodologies (Cockburn and Highsmith, 2001). When management provide the resources needed for using the agile methodology, software developers are more likely to adopt it.

**External support** refers to the use of external training and consultants (Roberts et al., 1998; 2001). Roberts et al. (1998) suggested that outside expertise was frequently required because, in many cases, no one in an organization had the expertise required to effectively implement a new SDM. Furthermore, internal trainers may not possess specialized skills and expertise to develop training courses and materials. In the domain of agile methodologies, Droka et al. (2004) suggested that a consultant who was outside an organization would bring a fresh perspective to the development team and ease the transition to agile methodologies. Similarly, Schatz and Abdelshafi (2005) suggested that it would be extremely helpful to get honest, objective feedback from an outside source. Therefore, software developers are more likely to adopt agile methodology when external support is provided.
Voluntariness is defined as “the extent to which potential adopters perceive the adoption decision to be non-mandatory” (Agarwal and Prasad, 1997; Hartwick and Barki, 1994; Moore and Benbasat, 1991). Voluntariness has been found to be a significant factor, with a direct effect on the intention to adopt SDMs (Hardgrave et al., 2003; Riemenschneider et al., 2002). When the usage of the agile methodology is not mandatory, individuals are less likely to adopt it, as switching to a new SDM is perceived as a radical change to the behavior processes in conducting one’s work. Therefore, software developers are less likely to adopt agile methodology when the use of agile methodology is voluntary.

Subjective norm is defined as the influence of important referents on an individual’s acceptance of an SDM (Hardgrave and Johnson, 2003). Previous studies found that subjective norm significantly affected intention (e.g. Taylor and Todd, 1995; Venkatesh and Davis, 2000). In particular, Venkatesh and Davis (2000) found that the effect of subjective norm on intention was moderated by experience and voluntariness. In the domain of SDMs, subjective norm was found to be an important determinant (Hardgrave et al., 2003; Hardgrave and Johnson, 2003; Riemenschneider et al., 2002), 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. Therefore, software developers are more likely to adopt agile methodology when subjective norm of using agile methodology is strong.

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” (McGehee and Thayer, 1961, p. 10). Since the adoption of a technological innovation creates uncertainty in the minds of potential adopters about its likely consequences (Rogers, 1995), 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 (2000) found that structured training positively affected individuals’ beliefs about perceived compatibility of an innovation; Riemenschneider and Hardgrave (2001) found that training had a positive effect on perceived ease of use of an SDM; Roberts and Hughes (1996) suggested methodology training was the key to successful SDM implementation. Therefore, software developers are more likely to adopt agile methodology when adequate training is provided.

Agile methodology characteristics
Perceived usefulness refers to the degree to which an individual expects that adopting an SDM will improve his or her individual job performance (Hardgrave et al., 2003). Previous studies on the acceptance of SDMs (Hardgrave et al., 2003; Riemenschneider et al., 2002) found that perceived usefulness was a significant factor in predicting the acceptance of SDMs. Consistent results were found in studies on other software process innovations such as programming languages (e.g., Agarwal and Prasad, 2000) and CASE tools (e.g., Iivari, 1996). Overall, prior research suggested that the more an innovation was perceived as enabling an increase in job performance, the more likely that it would be adopted.

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” (Moore and Benbasat, 1991, p. 197). It has been found to be an important factor affecting the intention to adopt an innovation in previous studies on various technologies (e.g., Davis et al., 1989). However, no direct effect of perceived ease of use was found in influencing user acceptance in the domain of SDMs (Riemenschneider et al., 2002). Such a distinction is possibly due to the difference
in the usage contexts of IT tools and SDMs, as SDM use is more radical and mandatory than tool use, and thus the relevance of how easy or how hard the behavior is to perform reduces (Riemenschneider et al., 2002). Moreover, since the SDM being tested in previous studies (i.e., Hardgrave et al., 2003; Riemenschneider et al., 2002) was based on the structured development paradigm which had been very popular in the software industry, perceived ease of use may not be a crucial determinant because developers may already be familiar with these traditional practices in the SDM and find them easy to use. On the other hand, the principles of agile methodologies are totally opposite to those of traditional SDMs, such as lessened reliance on documentation and increased acceptance of changes. Therefore, perceived ease of use is expected to have a greater effect when developers switch from traditional SDMs to agile methodologies than the case in which developers switch to another traditional SDM.

**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 (Hardgrave et al., 2003). 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. Therefore, perceived compatibility is expected to be an important factor in the domain of SDMs. Its importance has been justified in previous empirical studies on traditional SDMs (e.g., Hardgrave et al., 2003; Riemenschneider et al., 2002), and case studies on agile methodologies (e.g., Cockburn and Highsmith, 2001; McManus, 2003).

**Result demonstrability** refers to the degree to which an innovation is perceived to be amenable to demonstration of tangible advantages (Moore and Benbasat, 1991). Riemenschneider et al. (1998) found that result demonstrability was 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 most notable features suggested by agile methodologies are: 1) simple planning, 2) short development cycle, 3) earlier release, and 4) frequent customer feedback (Huo et al. 2004). These features allow developers to tell the results of using agile methodologies by performing frequent testing and receiving frequent customer feedback, resulting in higher result demonstrability of the agile methodology.

**Perceived maturity of technology** is a function of technology uncertainty and inexperience (Lee and Kim 1998). Cho and Kim (2001) 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.

**Customer relationships**

**Shared understanding** represents the extent to which the work values, norm, philosophy, problem-solving approaches, and prior work experience of a dyad are similar
Shared understanding removes the barriers to understanding and acceptance between a source and a recipient (Krauss and Fussell, 1990), and facilitates inter-firm knowledge transfer between a client and a consultant (Ko et al., 2005). This enables software developers and customers to work toward a common goal more effectively in agile development. Therefore, software developers are more likely to adopt agile methodology when they have good shared understanding with their customers.

**Arduous relationship** is defined as an emotionally laborious and distant relationship between a source and a recipient (Szulanski, 1996). 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. Therefore, an arduous developer-customer relationship will hinder agile development. Software developers are less likely to adopt agile methodology when they have arduous relationship with their customers.

**Discussion**
In the previous section, we have introduced a framework for examining individual acceptance of agile methodologies. The framework consists of four categories of factors, namely individual characteristics, organizational characteristics, agile methodology characteristics, and customer relationships. The influences of these factors have been justified by prior research and a reexamination of their applicability to the domain of agile methodologies. The framework could act as a basis for future empirical research, which is currently inadequate in the domain of agile methodologies (Erickson et al. 2005). At the same time, this framework could provide guidance in the process of introducing agile methodologies in practice, by identifying crucial aspects that practitioners should pay attention to.

Future theoretical development may be advanced by identifying crucial factors that have strong effect on the acceptance of agile methodologies. Previous studies on the acceptance of technology have suggested that characteristics of technology are significant determinants of the acceptance of various technologies or IT tools (e.g., Agarwal and Prasad, 1997; Chau, 1996). However, it is unclear whether characteristics of agile methods would still dominate in the domain of SDMs, because no existing study has empirically examined factors from all four categories in our proposed research framework. Moreover, Sultan and Chan (2000) pointed out that characteristics of technology may not be crucial when the subjects were experienced programmers. This justifies the need to re-examine selected factors from the four categories in our proposed research framework, in order to understand developers’ concerns about adopting agile methodologies from a broader perspective.

Researchers may also explore the indirect effects of certain factors in our proposed framework. Since our framework consists of a large number of potential factors, it is unlikely that all of them would have significant direct effects on the acceptance of agile methodologies. For example, perceived ease of use was found to have an indirect effect on the acceptance of SDM mediated through perceived usefulness (Hardgrave et al., 2003), whereas it did not have any direct effect on user acceptance (Riemenschneider et al., 2002). Furthermore, those constructs (e.g., perceived usefulness) in theoretical paradigms, such as TAM and DOI, differ from other factors (e.g., training) in terms of abstract level and are too general to provide insights for practical use of agile methods. For example, while perceived usefulness was found to be a significant determinant of the acceptance of SDM, little is known about what external factors contribute to a developer’s conclusion that an SDM is
useful to him/her. It would be fruitful to examine antecedents of general constructs such as perceived usefulness and perceived ease of use, in order to gain understanding in the basis on which developers form their expectations on particular characteristics of agile methodologies. For example, developers may think that agile methodologies are useful, because the organization has provided adequate training. In this case, our framework would assist in providing a number of potential antecedents related to various characteristics.

For practice, our proposed research framework has listed a number of potential facilitators and barriers to the acceptance of agile methodologies. When organizations decide to deploy an agile methodology, they may evaluate their decision from various aspects, i.e., the four categories of factors in our framework. It is inadequate to consider only the characteristics of the agile methodology while neglecting the others, as the four categories of factors are likely to be interrelated. For instance, without adequate training and external support, developers may have difficulties in understanding the characteristics of the agile methodology. IT managers may also concentrate on how to tailor developers’ views on agile methodologies to be deployed. By implementing facilitating conditions such as external support and training, organizations would be more effective in convincing their staff that they have adequate resources to use, or at least, to try the new agile methodology.

Conclusion
This article has proposed a research framework to understand individual adoption of agile methodologies. Our study contributes to the literature in two ways. First, we synthesize findings of previous studies on SDMs and re-examine the applicability of the identified factors to the domain of agile methodology. Second, we fill a gap in previous research by highlighting the important role of customers in agile development and introducing factors related to developer-customer relationships. The proposed framework may be used to understand how various categories of factors affect developers’ decision to adopt agile methodologies, and is amenable to empirical testing.

References


