pISSN 1976-913X
eISSN 2092-805X

J Inf Process Syst, Vol.9, No.1, March 2013 http://dx.doi.org/10.3745/JIPS.2013.9.1.069

Agile Software Development Framework in a Small Project Environment

Seiyoung Lee* and Hwan-Seung Yong**

Abstract—Agile methods are highly attractive for small projects, but no agile method works well as a standalone system. Therefore, some adaption or customization is always required. In this paper, the Agile Framework for Small Projects (AFSP) was applied to four industry cases. The AFSP provides a structured way for software organizations to adopt agile practices and evaluate the results. The framework includes an extended Scrum process and agile practices, which are based on agility and critical success factors in agile software projects that are selected from Scrum, XP, FDD, DSDM and Crystal Clear. AFSP also helps software managers and developers effectively use agile engineering techniques throughout the software development lifecycle. The case study projects were evaluated on the basis of risk-based agility factors, the agility of the adopted practices, agile adoption levels, and the degree of the agile project success. The analysis of the results showed that the framework used in the aforementioned cases was effective

Keywords—Agile Methods, Small Software Projects, Industrial Case Study

1. INTRODUCTION

Small enterprises (with fewer than 50 employees) and very small enterprises (VSEs with fewer than 25 employees) represent up to 85% of all software companies in countries that have an advanced IT industry [1, 2]. Generally, small companies are very responsive and flexible, and are directly/indirectly involved in many small projects. The challenges in managing small projects come from tight planning and responsiveness. On the other hand, the challenges in managing larger projects consist more of analyzing the complexity for business areas, technology, risks, and managing a large number of stakeholders. Thus, small projects require a different approach than larger ones.

There is a debate between proponents of traditional methods vs. agile methods regarding this approach. Traditionalists believe there is no need for a completely different method. They believe the use of a more streamlined version of traditional standard methods is more appropriate [1, 3, 4]. On the other hand, agile methods are specifically designed with small teams in mind and they have an entirely different approach to scaling up agile processes for large teams. Since the mid 1990's, agile methods have been suggested as a way of maximizing business value through small, self-organizing teams using flexible technologies and early customer involvement to iteratively improve software [6], especially in small software projects [7-10]. Lastly, Hybrid approaches that combine both methods are feasible and preferable [2, 11, 12].

Manuscript received February 17, 2012; first revision July 31, 2012; accepted October 12, 2012.
Corresponding Author: Seiyoung Lee
* Software Engineering Center, National IT Industry Promotion Agency, Seoul, Korea (sylee@nipa.kr)
** Department of Computer Science and Engineering, Ewha Womans University, Seoul, Korea (hsyong@ewha.ac.kr)

In this paper, we introduce an agile software development framework to assist software practitioners in developing small and medium-sized projects, the success of which is pointed out in several industry cases. The paper is organized as follows: Section 2 presents software engineering approaches in small settings and recent research on agile software development frameworks, along with our research objectives. Section 3 describes the Agile Framework for Small Projects (AFSP) and evaluates the degree of agility in the framework. In Section 4, we implement and validate the AFSP in four cases. Section 5 discusses the results of the case study and finally, Section 6 provides our concluding remarks and some directions for future research.

2. BACKGROUND

This section briefly reviews the following two key concepts: small software projects and agile methods. First, we describe an overview of small software projects and software engineering approaches in a small project environment. We then summarize the available agile research literature to justify the need for this study and then state the objectives of this research.

2.1 Small Software Projects

There are some factors that determine whether we should classify a project as small. A small project usually:

- Lasts six months or less in duration
- May involve part-time work
- Has ten or fewer team members
- Involves a small number of skill areas
- Has a single objective and solution that is readily achievable
- Has a narrowly defined scope and definition
- Affects a single business unit and has a single decision maker
- Doesn't require automated solutions from external project sources
- Has no political implications
- Produces straightforward deliverables with interdependencies among skill areas
- Has available funding.

Given this definition, Rowe described five challenges to small projects that make traditional approaches ineffective [3], such as:

- Lack of planning
- Low priority
- Inexperienced project teams
- Project manager responsible for multiple functions
- The use of standard project management tools and processes for small projects.

2.2 Software Engineering Approaches to Small Settings

In spite of the huge number of small projects around the world, there is a relatively small

amount of literature discussing software engineering solutions that is focused on only small projects. In this section, we classify software engineering approaches in a small environment into three categories: the streamlined approach, the agile approach, and the hybrid approach.

The first method of managing small projects is to scale down standard methods to meet a small project's needs. Rowe created a streamlined guideline for small projects. Even though the documents have been shortened, the underlying process is still the Project Management Body of Knowledge (PMBOK). ISO 9001 and the Unified Process are very rich and complex and are suited to the development of very large commercial software products [3]. Trengove and Dwolatzky developed a software engineering process based on these two frameworks, but did so in a way that is appropriate for small-scale software development projects [4]. From the perspective of software process assessment and improvement, to stay within the CMMI framework while minimizing assessment time and cost, some small companies run a class B or C assessment instead of a SCAMPI (Standard CMMI Appraisal Method for Process Improvement) class A assessment. Class B and C appraisals are scaled-down assessments that focus only on a single project or process area and don't produce any formal ratings [1]. Also Habra et al. suggested the OWPL gradual Software Process Improvement (SPI) approach as a way of initiating SPI in VSEs. This process involves a series of three assessments: Micro-Evaluation, OWPL evaluation and SPICE, and CMM or CMMI assessment [5].

The second method is an agile approach. Wang recommended agile methods for small- and medium-size software development, while recommending traditional methods for large-scale, highly complex and/or geographically dispersed projects [9]. Boehm also presented agile methods as extremely attractive for small projects, while plan-driven methods are most needed in high assurance software [10]. Rising and Janoff described Scrum as an appropriate development process for small teams with less certain requirements [7]. Furthermore, Pikkarainen proposed that the agile principles and practices be used as part of the assessment process and to use the generated information to improve the software development process [10].

Lastly, Boehm and Turner argued that agile and plan-driven methods each have their own area of strength. They also emphasized balance and attempt to make a case for hybrid strategies [11]. Cohen et al. also argued that agile and traditional methods have a symbiotic relationship, in which factors such as the number of people working on a project, application domain, criticality, and innovativeness will determine which process are chosen [12]. In the meantime, Caffery et al. created the Adept assessment method, which combines the CMMI Class C appraisal guidelines and an adapted agility/discipline assessment approach in a unified model. This cohesive model covering both plan-driven and agile approaches is tailored for small software companies [2].

2.3 Agile Methods and Small Projects

In recent years, agile methods in general, and Scrum and XP in particular, have gained increasing popularity around the world, and have proven to be effective [13-19]. Agile methods are a lightweight alternative to traditional methods, and are based on the four broad processes of using iterative development, customer feedback, small software development teams, and flexible software technologies.

Some of the better-known agile methods include Extreme Programming (XP), Scrum, Feature Driven Development (FDD), Dynamic Systems Development Method (DSDM), Lean Development, Crystal Methods, and Adaptive Software Design (ASD). All agile methods have com-

mon values, but each approach differs in execution.

For example, ASD is the most abstract while Crystal proposes concrete practices that have been adapted to projects of differing size. XP and Crystal propose practices for the whole software lifecycle, while the others are only frameworks or cover only a part of the lifecycle. XP concentrates on the development rather than on the managerial aspects of software projects. On the other hand, DSDM and Scrum focus on project management and are agnostic about the underlying technical approach. Both emphasize business value, fixed time boxes, significant customer involvement, and the development of high priority items first [20]. Scrum and XP practices are more or less disjointed and are often used together to increase productivity and quality [17, 21, 22]. Agile methods are more suitable for small teams [8, 9, 12], but also have been used successfully in large projects and distributed teams [18, 21].

2.4 Empirical Evaluations of Agile Software Projects

Agile methods are effective and suitable for many situations and environments [11, 23, 24], but very few empirically validated studies support the claims made by the agile proponents [15, 25]. Recently, there have been some studies on the evaluation of the effectiveness of using agile methods. Boehm and Turner provided a way to assess how effective agile or plan-driven methods will be for a particular project. This model keeps focus on the assessment of the agile and plan-driven risks rather than on finding weaknesses and strengths of the practice. The risk-based agility assessment model provides comparable facts for agile project evaluation [11]. Qumer and Henderson-Sellers defined agility as follows:

"Agility is a persistent behavior or ability of a sensitive entity that exhibits flexibility to accommodate expected or unexpected changes rapidly, follows the shortest time span, uses economical, simple and quality instruments in a dynamic environment and applies updated prior knowledge and experience to learn from the internal and external environment."

They developed a 4-Dimensional Analytical Tool (4-DAT) for the assessment of the degree of agility, applied the tool to six agile methods and two traditional methods, and compared the results. 4-DAT facilitates the examination of agile methods from four perspectives: 1) method scope characterization, 2) agility characterization, 3) characterization of agile values, and 4) software process characterization. Such an evaluation can be useful when selecting an appropriate agile method and/or practices for a particular project [26].

Qumer and Henderson-Sellers also proposed an Agile Adoption and Improvement Model (AAIM), which assists software organization that is in transition to an agile development environment. The AAIM includes six agile stages, which are embedded in the three agile blocks of: prompt, crux, and apex. Each block and stage has a name and specifies the agile practices to follow, in order to achieve the particular AAIM level (AAIML). This model also can be used for the assessment of a specific level of agility adoption and to advance to the next level [27].

Chow and Cao identified six Critical Success Factors (CSFs) for agile software development projects and summarized their attributes. A survey was conducted among agile professionals who gave survey data from 109 agile projects from 25 countries around the world. The collected data were analyzed using the multiple regression method. The analysis results revealed that the CSFs are found to be: 1) delivery strategy, 2) agile software engineering techniques, 3) team capability, 4) project management process, 5) team environment, and 6) customer involvement [28].

2.5 Objectives of This Research

Small projects are not just scaled-down versions of large ones, and are not necessarily easier to manage than large projects. They may still be complex and involve a range of departments, resources, suppliers, and customers to complete them.

Agile software development has had a huge impact on how software is developed worldwide [13, 14, 19]. However, even though there are many agile methods, which are known to be effective for small projects [7-9], little is known about practical adoption solutions and how results are evaluated. Agile is an umbrella concept encompassing many different methods and each method uses its own vocabulary and terminology. Because projects differ in their scale, scope, and technical challenges, one method does not suit all circumstances. Thus, software practitioners need to carefully select and apply the most beneficial agile practices for their particular project requirements.

The purpose of this study is to provide a structured way for software organizations to adopt agile practices and evaluate the results, especially in a small project environment. Here, we introduce AFSP, an open framework that integrates advantages from different agile methods and different evaluation criteria. In order to prove the efficiency of our framework, we first show AFSP with a higher degree of agility than a standalone agile method (Section 3.4) And then, we applied AFSP to four industry cases (Section 4) and analyzed the results of the case study (Section 5).

3. AGILE FRAMEWORK FOR SMALL PROJECTS

The main challenge for the organization is how to apply the most suitable agile practices to its specific product development context [29]. The Agile Framework for Small Projects (AFSP) uses Scrum as the backbone and provides a structured process that guides software organizations in adopting agile practices. In this section, we provide both a general overview of AFSP and step-by-step instructions. We then explain the process and practice pool, and evaluate the degree of agility in the framework.

3.1 AFSP Overview

The framework consists of two main components: 1) the AFSP process, which helps to build the correct products; and 2) the AFSP practice pool, which helps to determine the techniques and tools to build the products correctly. Also, the AFSP incorporates the following four evaluation tools from the agile literature: 1) the risk assessment model [11], which can be used as a starting point for agility evaluation and agile project selection; 2) 4-DAT [26], which can be used to measure the agility of agile practices; 3) AAIM [27], which can be used to determine an agile adoption or improvement level, or to assess a specific level of agility adoption and to advance to the next level; and 4) CSFs [28], which can be used to measure the degree of agile project success.

Step 1: *Identify the five risk-based agility factors.*

First, the project team identifies the five risk-based agility factors affecting the selection of either agile or plan-driven methods: 1) size: the number of people on the team, 2) criticality: prod-

ucts safety criticality, 3) dynamism: degree of requirements and change in technology, 4) personnel: the skill and experience of the team, and 5) culture: the support for agile software development that is provided by an organization.

This step helps to enhance the ability of project team members to understand their environment and their organizational capabilities and to identify and collaborate with the project's stakeholders. Section 4.2 illustrates the details of the risk assessment model.

Step 2*: Determine the final set of agile practices.*

Next, the project team determines agile practices where their strengths can be best applied and where their risks can be minimized. The AFSP offers a pool of agile practices, based on agility and CSFs in agile software projects that are selected from Scrum, XP, FDD, DSDM, and Crystal Clear. Also, the 4-DAT can be used to measure the agility of each practice as needed. Section 3.3 presents the details of the recommended practices provided by AFSP and Section 4.3 shows the industry cases.

Step 3*: Project build and deployment.*

In this step, the project team focuses on project development and deployment by following the AFSP process as it has been incorporated into the agile practices as determined in Step 2, from project beginning to end. The specifics of the process will depend primarily on the developer organization's capabilities and experience in the application area. The management team must constantly monitor the performance of its selected practices while keeping an eye on the environment. Section 3.2 presents the details of the AFSP process and Section 4.3 shows the industry cases.

Step 4*: Assess an agile adoption or improvement level.*

The project team assesses an AAIML within a software development organization. The AAIM provides six agile levels: 1) AAIML 1, which introduces and establishes the basic agile properties (speed, flexibility, and responsiveness) in a software development process; 2) AAIML 2, which enables communication and collaboration among the people by establishing good communication and cooperation protocols; 3) AAIML 3, which emphasizes the production of the executable artifacts with minimal documentation; 4) AAIML 4, which emphasizes the value of people without ignoring the importance of the software development tools and processes; 5) AAIML 5, which focuses on the establishment of a learning environment; and 6) AAIML 6, which focuses on the establishment of a lean production environment (quality production with minimal resources and within a minimum timeframe) and maintains agility in the process as a focal point.

This step can be used to assess how effectively the agile process and practices are performed within an organization. The adoption model can also help to establish a systematic agile software development environment. Section 4.4 illustrates the application of AAIM in the four industry cases.

Step 5*: Evaluate the project success.*

Finally, the project team can quantitatively measure the Degree of Agile Project Success (DAPS) based on the survey study of CSFs in agile software projects. The CSFs can be defined as the factors that must be present for the agile project to be successful. The DAPS shows how

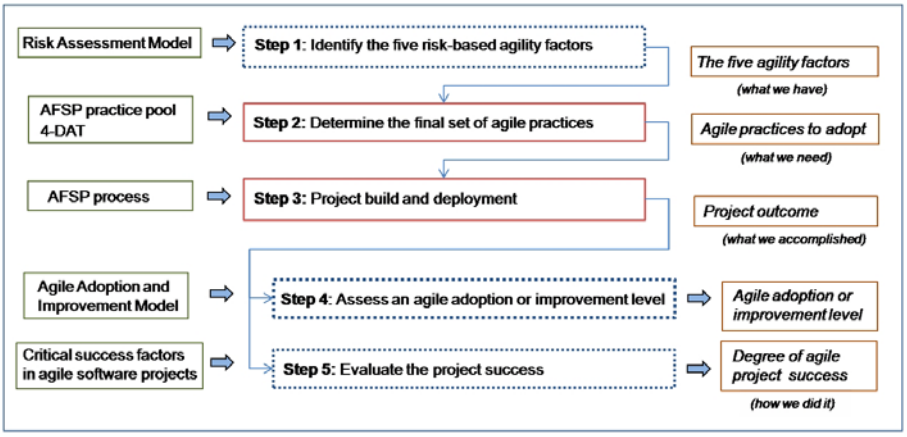

Fig. 1. The five steps in the application of the Agile Framework for Small Projects

well the project organization performs to achieve success. This step helps to identify and measure an organization's performance and Section 4.5 shows how to calculate the DAPS.

Fig. 1 summarizes the five steps in the application of AFSP to software projects. The five steps being: what we have, what we need, what we accomplished, and how we did it. All the evaluation steps, including Steps 1, 4, and 5, are optional, and each of the steps is presented in detail in Section 4, along with the industry examples.

3.2 The AFSP Process

The AFSP process encompasses the entire software development lifecycle, and incorporates the best practices to make it work. The process uses a great deal of Scrum components. Scrum is an agile project management method and does not propose how the product should be engineered. Due to its simplicity Scrum has become the most popular agile method to be put into use lately [13, 16, 19, 30].

Scrum can be implemented on top of existing software engineering processes, and allow us to put some lightweight, structured processes into place quickly [7, 17, 21]. On the other hand,

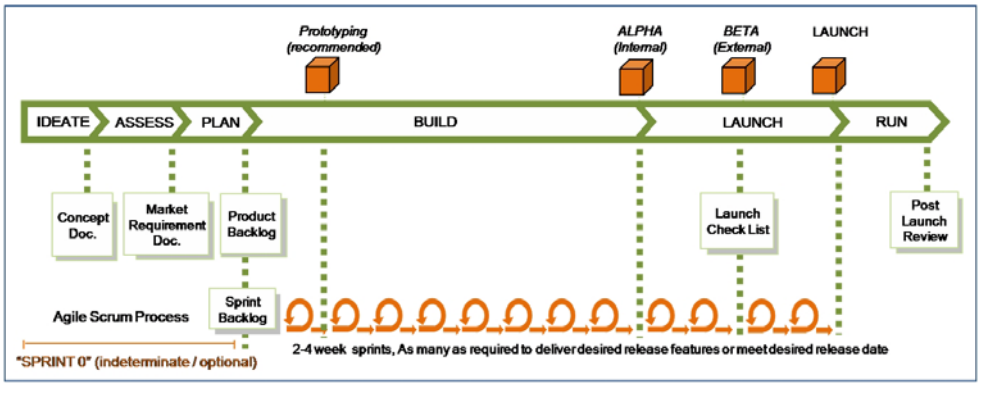

Fig. 2. The AFSP process incorporated with Scrum

"Sprint 0" in Fig. 2 can be an option in the AFSP process. It could be somewhat of a bridge between Scrum and the traditional plan-driven method. This indeterminate length allows for some of the more traditional planning, specification, and design activities to be completed as needed. Although it is a non-agile phase, there are definite benefits to this concept, which could prove to be advantageous for the product in the long run.

Scrum projects make progress in a series of iterations called "sprints," which are typically 2-4 weeks in length. The product requirements exist in a "product backlog" that is prioritized by the product manager. The team selects the highest priority items from the product backlog that they can get done in a sprint. The product is designed, coded, and tested during the sprint. The team meets daily to review the progress and demonstrate what they have built at the end of a sprint. Also, the team conducts a "retrospective" to help them improve the process after every sprint.

3.3 The AFSP Practice Pool

The purpose of this section is to enable software practitioners to choose and apply the right agile practices at the right time. It should be noted that small projects, which are more common than the large ones, require less effort. Software development practices take place at three different levels: the organizational development level, the project management level and the software engineering level [31]. We first classified agile practices from Scrum, XP, FDD, DSDM, and Crystal Clear into each of these levels as based on the agile method literature. We used opinions of domain experts and CSFs in agile software projects as the criteria to eliminate duplication and to better categorize the practices into the software development lifecycle.

We also measured the agility of the practices using 4-DAT and removed some practices with less degree of agility. Finally the candidate practices for the AFSP practice pool were identified as shown in Fig. 3, and a relatively large amount of practices from Scrum and XP have been included as a result. Scrum focuses more on management techniques, whereas XP is geared more towards development techniques. Thus, they are very complementary and also integrate

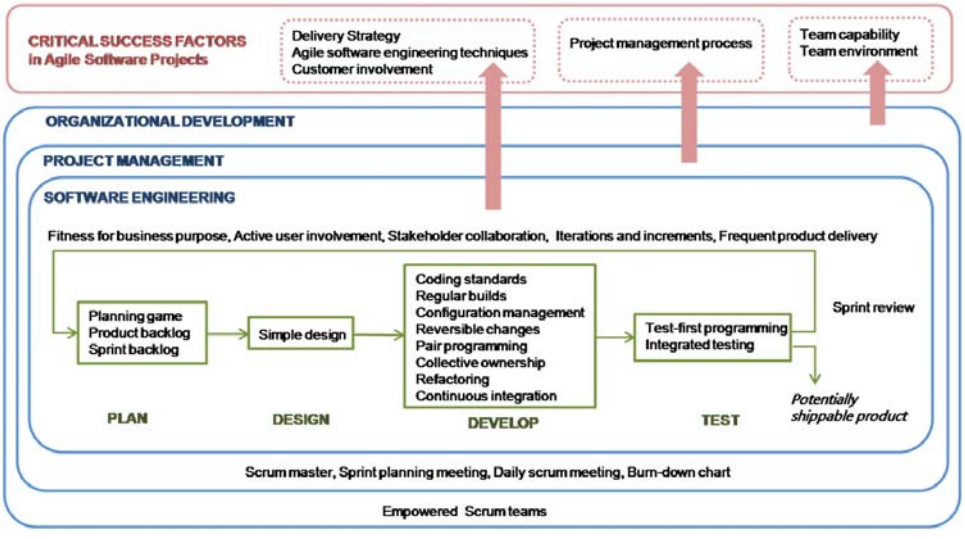

Fig. 3. The AFSP practice pool encompassing CSFs in agile software projects

together quite nicely because of the sharing of some core practices. The AFSP pool practices can be selectively incorporated into the AFSP process based on project requirements and developmental factors. The decision about what level of engineering is appropriate depends on the context, and on an organization's readiness.

3.4 Evaluating the Degree of Agility in the AFSP

According to some literature and our experience, more agile practices are more effective for small-sized projects. This is also based on the following definition from Qumer and Henderson-Sellers [26]:

"A software development method is said to be an agile software development method when it is people focused, communication-oriented, flexible (ready to adapt to expected or unexpected change at any time), speedy (encourages rapid and iterative development of the product in small releases), lean (focuses on shortening timeframe and cost and on improved quality), responsive (reacts appropriately to expected and unexpected changes). In addition, such a method should provide an environment for learning (focuses on improvement during and after product development)."

We then tested the AFSP using 4-DAT, which provides a mechanism to quantitatively measure the Degree of Agility (DA) of a software development method at a specific phase in the process and with a specific practice. Eq. (1) was used to check the existence of agility in the AFSP at both a process level and practices level where Object = {Method, Framework}:

$$DA(Object = 1/2(DA(Object, Phases) + DA(Object, Practices)) \quad (1)$$

An object has a process (PS) and one or more practice(s) (PR), whereby each process has different phases (PH). The degrees of agility in a process (phases) and practices are calculated in terms of the five features of agility for each phase of a process and for each practice, such as: flexibility (FY), speed (SD), leanness (LS), learning (LG), and responsiveness (RS). As shown in Table 1, values of 0 or 1 were entered for each phase and practice. The overall phase/practice totals and the average degree of agility of each process/practice level were calculated.

Agility of PHi or PRi = {FY + SD + LS + LG + RS}
*DA(Object, Phases) = (Total Agility in PH) / (n * 5)*
*DA(Object, Practices) = (Total Agility in PR) / (m * 5)*

where n is number of phases in a process, m is number of practices, and the constant 5 represents the five agility attributes. The DA(Object) is a value between 0 and 1, and the threshold of agility is between 0.5 and 0.6 for any software development method.

The literature pertaining to the specific agile method was used as the source of the numerical inputs for the analysis. In addition, we compared some results of the AFSP with the values of DA for the five agile methods that were evaluated in [26], as shown in Table 2. The result of DA(AFSP, Phases) is equal to the result of DA(Crystal, Phases), which was evaluated as the most agile at the process leve1. Furthermore, DA(AFSP, Practices) has a higher agile figure than the DA(Scrum, Practices), which was evaluated as the most agile at the practice level. Therefore,

Table 1. Degree of agility in the AFSP (FY: flexibility, SD: speed, LS: leanness, LG: learning, RS: responsiveness)

AFSP	Agility Features					Total
	FY	SD	LS	LG	RS	
Phases of the Process (PHi)						
Sprint planning	1	1	0	1	1	4
Build	1	1	0	1	1	4
Launch	1	1	0	1	1	4
Total	3	3	0	3	3	12
DA(AFSP, Phases)	3/3	3/3	0/3	3/3	3/3	12/(3*5)
Practices (PRi)						
Planning game	1	1	0	1	1	4
Product backlog	1	1	0	1	1	4
Sprint backlog	1	1	0	1	1	4
Simple design	1	1	1	1	1	5
Coding standards	1	1	1	1	1	5
Regular builds	1	1	0	1	1	4
Configuration management	1	1	0	1	1	4
Reversible changes	1	1	0	1	1	4
Pair programming	1	0	0	1	1	3
Collective ownership	1	0	0	1	1	3
Refactoring	1	1	1	1	1	5
Continuous integration	1	1	1	1	1	5
Test-first programming	1	1	0	1	1	4
Integrated testing	1	1	0	1	1	4
Sprint review	1	1	0	1	1	4
Empowered Scrum teams	1	1	0	1	1	4
Scrum master	1	1	0	1	1	4
Sprint planning meeting	1	1	0	1	1	4
Daily scrum meeting	1	1	0	1	1	4
Burn-down chart	1	1	0	1	1	4
Total	20	18	4	20	20	82
DA(AFSP, Practices)	20/20	18/20	4/20	20/20	20/20	82/(20*5)

Table 2. Degree of agility in the 5 agile methods [26] and AFSP – overall comparison

Objects	*DA(Object, Phases)*	*DA(Object, Practices)*	*DA(Object)*
XP	0.702	0.732	0.717
Scrum	0.603	0.800	0.702
FDD	0.485	0.703	0.594
DSDM	0.466	0.684	0.575
Crystal	0.800	0.732	0.766
AFSP	0.800	0.820	0.810

DA(AFSP) which is the average of DA(AFSP, Phases) and DA(AFSP, Practices) has a higher agile figure than what would be expected from a standalone agile method. This is due to the inclusion of the most agile practices in the AFSP.

4. IMPLEMENTATION AND VALIDATION OF THE AFSP

A number of assessment and validation methods have been applied to carry out the tests on AFSP. We also actually implemented four case projects from Yahoo! USA, Spain, Korea, and Taiwan, on the basis of the AFSP. The focus of this case study research was to empirically validate the applicability of AFSP to support the adoption and evaluation of agile approaches.

4.1 Overview of the Four Case Projects

In this section, we introduce our industry case study. The first case is a redesign of the 38 international versions of the news search pages. The project was completed by an international software development team in Sunnyvale, California, which included senior developers, who were responsible for the modeling and implementation of the new product-enhancement features using this agile approach, and the product manager, who was responsible for the prioritization of developing product features.

The other three cases are localization projects that were part of the My Yahoo! globalization project, which was hosted by a global service group in the US [32]. The Spanish team conducted the second case project, while the Korean and Taiwanese teams conducted the third and fourth ones, respectively. Each international version of the My Yahoo! service is an independent product with consistent features that are based on a single global codebase. Therefore the international teams were able to work mostly independently once they understood the overall vision/strategy, received proper training/coaching, and acquired systems that were initiated by the US group.

All of the project teams used the corporate Wiki system to manage the Scrum backlogs, teams, and products. They also produced current statuses and information using simple web page for-

Table 3. Development factors in the case projects

Context Factory	Case 1	Case 2	Case 3	Case 4
Team Size	6	5	5	5
Location	Sunnyvale, CA	Spain	South Korea	Taiwan
Domain Expertise	High	High	High	Middle
Domain	Web application	Web application	Web application	Web application
User Stories	28	45	57	49
Nature of Projects	Enhancement	Localization	Localization	Localization
Relative Complexity	Moderate	Moderate	Moderate	Moderate
Person Months	12	18	24	12
Elapsed Months	4	6	8	3
Language	PHP, C++, JS	PHP, C++, JS	PHP, JS	PHP, JS
Communication	Face to face, Call, Email, IM, Bugzilla, Wiki, Video Conference, WebEx			
Release Length	3 weeks	2 weeks	2 weeks	2 weeks

mats, and sometimes used photos and/or bios to emphasize the human element. Stakeholders were able to see user stories and backlogs and the status of the project in any desired detail. The corporate Bugzilla and CVS were used as a defect/task tracking system and software source code control system. The teams easily shared progress/status and tried to keep shared information constantly updated. Table 3 summarizes each project with the context factors included.

4.2 Identifying the Five Risk-based Agility Factors

Based on Step 1 in Section 3.1, we used the risk assessment model [11] as a starting point for agility evaluation even though this step is not necessary for most small projects. In this assessment model, the five axes are labeled based on the critical factors that describe a project environment (Fig. 4).

First, Size and Criticality represent the factors that Cockburn uses to distinguish between the lighter-weight Crystal methods toward the graph's center and the heavier-weight Crystal methods that appear toward the periphery. The Culture axis reflects that agile methods will succeed better in a culture that "thrives on chaos" than in one that "thrives on order," while the opposite is true for plan-based methods. For Dynamism, agile methods work well with both high and low change rates, but plan-driven methods work best with low change rates. The Personnel scale refers to the extended Cockburn method skill rating scale. For example, Level 1B people function well in performing straightforward software development in a stable situation, but will likely slow down an agile team that is trying to cope with rapid change, particularly if they form a majority of the team. Level 2 people are able to function well in managing a small, precedented, agile or plan-driven project but need the guidance of Level 3 people on a large or unprecedented project. Some Level 2s can become Level 3s with experience. So, agile methods require a richer mix of higher-level skills, while plan-driven methods can work well with both high and low skill

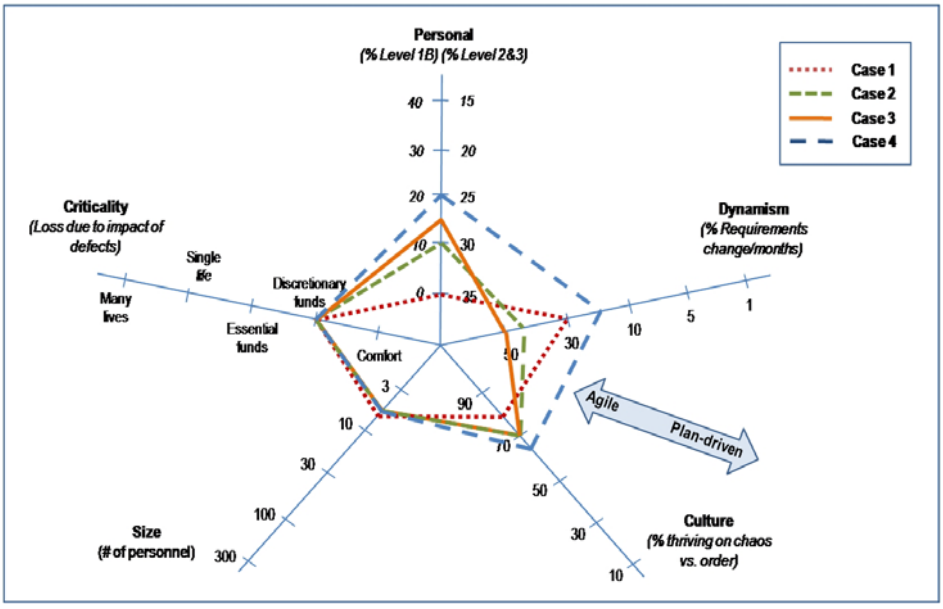

Fig. 4. The five risk-based agility factors of the four case projects

levels [33].

When the data points of a project for each factor are joined and the resulting shape is located directly around the center, this suggests the use of an agile method. Shapes that gather distinctly toward the periphery suggest the use of a plan-driven method. More varied shapes suggest the use of a hybrid method including both agile and plan-driven practices. As shown in Fig. 4, the four case projects were classified as “agile” because most of the data points are toward to the center of the graph. If the shape indicates that a hybrid “partially plan-driven, partially agile method” is appropriate, a risk-based strategy to incorporate both agile and plan-driven approaches should be considered.

4.3 Project Build and Deployment in the AFSP

The case project teams defined their own agility by determining practices to adopt using the AFSP practice pool. The first project was conducted using all of the agile practices from the AFSP practice pool to deliver greater business value and reduce organizational waste, as this team was more experienced in the use of agile methods.

On the other hand, the second to fourth projects were conducted with only thirteen practices in order to focus on the effective completion of localization work, such as the product backlog, sprint backlog, coding standards, regular builds, configuration management, reversible changes, continuous integration, test-first programming, integrated testing, sprint review, empowered scrum teams, scrum master, and sprint planning meetings. After the project kick-off, each international team worked closely with the global service group for a couple of weeks. During this period, the global service group provided an up-to-date localization manual, checklist, training, and coaching as needed. The global service group used the Scrum of Scrums [17] for upper-level coordination of multiple backlogs but didn't use it by the book. They adopted a “go-local” rule and allowed each international team to conduct their meetings at the most convenient time [32].

All of the project teams also used “Sprint 0” in the planning phases. With Sprint 0, the teams consolidated knowledge and developed their product vision and strategy by defining their product goals, features/functions, financial objectives, market opportunities, industry trends, and competitive differentiators. In subsequent sprints, they planned the next sprint, developed/localized the product, and validated it with users, while iterating as needed.

4.4 Assessing an Agile Adoption or an Agile Improvement Level

The AAIM [27] can be used in the planning phase to determine a target level of an agile adoption or improvement within a software development organization. However, in this paper, we used this adoption model as an evaluation criterion for the case study because the assessed level of an agile adoption can indicate how well the AFSP process and practices were followed within the project organization.

The first case project was a good candidate in testing the AFSP. The team was composed of a small set of senior engineers, a program manager/QA, and a product manager who were well trained and motivated. The project criticality and complexity were relatively moderate. Even though the product manager was located in Santa Monica, while all of the others were in Sunnyvale, California, there was no significant problem due to the separation of the team as the product manager was able to attend most of the sprint meetings in Sunnyvale. On the other hand, the

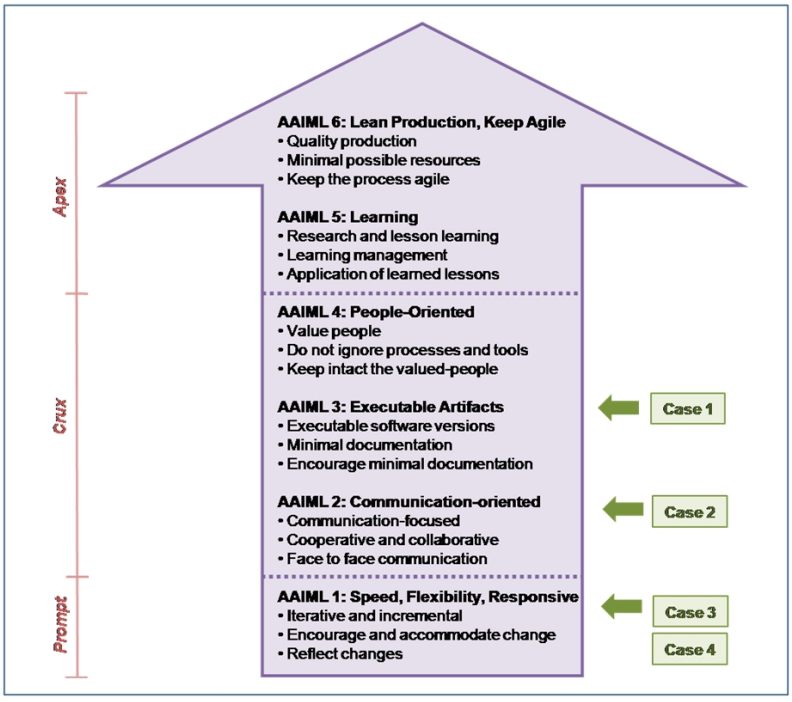

Fig. 5. Agile adoption and improvement levels for the case study (adapted from [27])

other projects were conducted with international teams that were not or less familiar with agile methods at that point. Only the Spanish team was better trained/experienced with agile practices than other international teams as part of the European service group. Therefore, most of the team members received early coaching and training from the corporate Agile group and learned how to facilitate key events in the Scrum before these projects started. These key events included iteration planning and retrospectives, attending daily stand-up meetings, and working with key team members to help answer questions and provide guidance.

As a result, it turned out that the first case project was engineered by focusing on AAIML 3, while the second one was performed by focusing on AAIML 2. For the third and fourth case projects, each team focused on AAIML 1 practices to iteratively carry out a systemic localization. Fig. 5 summarizes AAIM levels for the case study.

4.5 Evaluating the Success of the Project

In this section, we propose a simple equation to calculate the quantitative degree of agile project success, based on the survey study of CSFs in agile software projects [28]. The CSFs can be defined as the factors that have to be present for the agile project to be successful. The Degree of Agile Project Success (DAPS) represents how well the organization performs on the project to achieve success. Therefore, we used the value of DAPS as another evaluation criterion for the case study because the value can indicate how well the AFSP worked in each case project. Section 5.2 further discusses the evaluation criteria. The DAPS (Project) is a value between 0 and 1 and is calculated by Eq. (2):

Table 4. Attributes of the 6 CSFs in agile software projects [28]

Rank	Success Factor	Attributes
1	Delivery strategy	DS1: Regular delivery of software
		DS2: Delivering the most important features first
2	Agile software engineering techniques	AT1: Well-defined coding standards up front
		AT2: Pursuing simple design
		AT3: Rigorous refactoring activities
		AT4: Right amount of documentation
		AT5: Correct integration testing
3	Team capability	TC1: Team members with high competence and expertise
		TC2: Team members with great motivation
		TC3: Managers knowledgeable in agile methods
		TC4: Managers who have an adaptive management style
		TC5: Appropriate technical training for the team
4	Project management process	PM1: Following the agile-oriented requirement management process
		PM2: Following the agile-oriented project management process
		PM3: Following the agile-oriented configuration management process
		PM4: Good progress tracking mechanism
		PM5: Strong communication focus with daily face-to-face meetings
		PM6: Honoring a regular working schedule
5	Team environment	TE1: Collocation of the whole team
		TE2: Coherent, self-organizing teamwork
		TE3: Projects with a small team
		TE4: Projects with no multiple independent teams
6	Customer involvement	CI1: Good customer relationship
		CI2: Strong customer commitment and presence
		CI3: Customer having the full authority

$$DAPS(Project) = \sum_{i=1}^{m} W_i \times \frac{Sum_i}{N_i} \quad (2)$$

where *m* is number of CSFs (CFS) and *CFS = {DS, AT, TC, PM, TE, CI}* as shown in Table 4. *N* is the number of attributes in each *CFS*, *T* is the total of the value of attribute, and the constant *W* represents the weight of each CFS. The grading of each of the 25 attributes from Table 4 was evaluated on a dichotomous scale ("pass" or "fail") at each project team retrospective meeting.

The assessment results are shown in Table 5, in which a "1" indicates "yes" (or OK) to the question, while "0" indicates "no" (or not OK). Finally, the degrees of success of the case projects are as follows:

*DAPS(Case 1) = 0.3 * (2/2) + 0.2 * (5/5) + 0.2 * (5/5) + 0.1 * (6/6) + 0.1 * (2/4) + 0.1 * (3/3)*
*DAPS(Case 2) = 0.3 * (2/2) + 0.2 * (4/5) + 0.2 * (5/5) + 0.1 * (5/6) + 0.1 * (1/4) + 0.1 * (2/3)*
*DAPS(Case 3) = 0.3 * (2/2) + 0.2 * (4/5) + 0.2 * (3/5) + 0.1 * (5/6) + 0.1 * (1/4) + 0.1 * (2/3)*
*DAPS(Case 4) = 0.3 * (2/2) + 0.2 * (5/5) + 0.2 * (4/5) + 0.1 * (6/6) + 0.1 * (2/4) + 0.1 * (2/3)*

Table 5. Degrees of agile project success for the 6 CSFs in the case study

Success Factor	Weight	Attributes	Case 1	Case 2	Case 3	Case 4
Delivery strategy	0.3	DS1	1	1	1	1
		DS2	1	1	1	1
		Sum	2	2	2	2
Agile software engineering techniques	0.2	AT1	1	1	1	1
		AT2	1	1	1	1
		AT3	1	0	0	1
		AT4	1	1	1	1
		AT5	1	1	1	1
		Sum	5	4	4	5
Team capability	0.2	TC1	1	1	1	1
		TC2	1	1	0	1
		TC3	1	1	0	0
		TC4	1	1	1	1
		TC5	1	1	1	1
		Sum	5	5	3	4
Project management process	0.1	PM1	1	1	1	1
		PM2	1	1	1	1
		PM3	1	1	1	1
		PM4	1	1	1	1
		PM5	1	0	0	1
		PM6	1	1	1	1
		Sum	6	5	5	6
Team environment	0.1	TE1	0	0	0	0
		TE2	1	0	0	1
		TE3	1	1	1	1
		TE4	0	0	0	0
		Sum	2	1	1	2
Customer involvement	0.1	CI1	1	1	1	1
		CI2	1	1	1	1
		CI3	1	0	0	0
		Sum	3	2	2	2
DAPS(Project)			0.95	0.835	0.755	0.877

Based on our quantitative evaluation of the case projects, we can offer a threshold value of around 0.7-0.8 for any agile software projects that are likely to be successful [32]. Eventually, the four case projects were successfully built and deployed on target and the degree of agile project success in Table 5 supports these results.

5. DISCUSSION

We will now answer challenges that face small projects. The next subsection discusses a list of criteria that were used in evaluating the case projects.

5.1 Answers to the Challenges of Small Projects

We were able to address all of the challenges that face small projects in Subsection 2.1 due to the results of the application of the AFSP. First of all, to remedy the lack of planning, the AFSP includes a planning game, a sprint planning, and sprint '0' for simple but effective planning. One or a combination of all three can be used at the discretion of the project team. In addition, a risk assessment model, the AAIM or 4-DAT, can also be used in the planning phase.

Second, to address the issue of low priority and inexperienced project teams, the AFSP process is incorporated with Scrum. Scrum mechanisms exist to keep teams functioning at a high level. Iterative releases drive faster ROI (Return On Investment) and teams feel empowered and enthusiastic as productivity increases. Moreover, agile engineering practices provide an opportunity for team members to learn from each other by encouraging the team to work closely together.

Third, to tackle the situation where project managers are responsible for multiple functions, Scrum introduces a new management role, called “Scrum Master.” To ensure team empowerment, the Scrum Master works diligently on removing impediments, championing agile values, protecting the team from outside interference, and works as a facilitator and consensus builder. For small projects, one Scrum Master could manage several projects at the same time, rather than working within the team.

Lastly, for the use of standard project management tools and processes for small projects, the AFSP is open to the incorporation of best practices from agile methods that have been proven to be effective, especially for small projects [7-9].

5.2 Case Study Evaluation

This research seeks to provide an open framework for more effective agile software development. AFSP consists of the process, practice pool, and four evaluations tools. In order to apply AFSP to each case, we first assessed the risk-based agility factors of the project to identify the developmental factors. We then determined which agile practices to adopt from the AFSP practice pool based on the risk analysis results. After each project was completed and deployed with the use of the AFSP process incorporated with the practices, the agile adoption level of the project organization and the degree of the agile project success were evaluated.

We used two evaluation criteria to assess the case study, as mentioned in Section 4. First, all of the project organizations adopted the AFSP as the recommended approach and this showed that the AFSP could be easily adopted on the basis of the assessment results (Section 4.4). Second, all of the projects were completed successfully, as the DAPS values shows (Section 4.5). As a result, the case study organization has applied the AFSP to the rest of the My Yahoo! localization projects. In addition, the informal survey of international teams showed about a 30% improvement in the productivity and the satisfaction of the organization with the new localization process based on agile methods [32], and this outcome is consistent with the results of surveys conducted by the Yahoo! Agile group [30].

6. CONCLUSIONS

The best way to improve software productivity and quality is to focus on people. Agile soft-

ware development method is people-centric, communication-oriented, and willing to adapt to expected or unexpected changes at any time. It promotes the incremental and iterative development of the software in small releases, and focuses on reducing costs and improving quality. The Agile Framework for Small Projects (AFSP) addresses specifics regarding the application of agile techniques and the evaluation of the results.

In this paper, four industry cases were implemented and deployed successfully based on the AFSP. The analysis of the results indicated that the framework displayed a high degree of efficiency. AFSP is designed based on the comparison and analysis of the five agile methods including Scrum, XP, FDD, DSDM, and Crystal Clear. We also looked at the agility of the process and practices and the CSFs in agile software projects. The AFSP process also optionally accommodates the "big design up front" character from traditional methods. The AFSP practice pool offers general agile practices that can be used for any type of project.

The following four useful evaluation tools were added to AFSP: 1) the risk assessment model [11] to identify developmental factors and to build a balanced strategy including agility and discipline; 2) 4-DAT [26] to quantitatively measure the agility of a software development process and practices; 3) the AAIM [27] to determine an agile adoption or improvement level for software organizations, and to set up a systematic agile software development environment; and 4) the CSFs in agile software projects [28] to identify and measure a software organization's performance. For the first time, we have developed an empirical equation based on the survey study of CSFs in agile software projects to quantitatively estimate the degree of the success of an agile project.

The major contribution of this research is to provide a focus for management when they start adopting agile methods in their software projects. This paper focused agile methods on a small-project environment with small teams and less certain requirements. Agile methods look simple but there is a lot to learn. Working with experienced coaches and observing other teams is important for success. Therefore, we intend to add distributed software development strategy and a training-coaching model to the framework in the near future.

REFERENCES

[1] I. Richardson, C.G.v. Wangenheim, "Why Are Small Software Organizations Different?", IEEE Software, Vol.24, No.1, 2007, pp.18-22.

[2] F.M. Caffery, P.S. Taylor, G. Coleman, "Adept: A Unified Assessment Method for Small Software Companies", IEEE Software, Vol.24, No.1, 2007, pp.24-31.

[3] S. Rowe, *Project Management for Small Projects*, Management Concepts, USA, 2006.

[4] E. Trengove, B. Dwolatzky, "A software development process for small projects", International Journal of Electrical Engineering Education, Vol.41, No.1, 2004, pp.10-27.

[5] N. Habra, S. Alexandre, J-M. Desharnais, C.Y. Laporte, A. Renault, "Initiating software process improvement in very small enterprises", Information and Software Technology, Vol.50, No.7-8, 2008, pp.763-771.

[6] "Agile Manifesto", 2001, http://www.agilemanifesto.org

[7] L. Rising, N.S. Janoff, "The Scrum Software Development Process for Small Teams", IEEE Software, Vol.17, No.4, 2000, pp.26-32.

[8] B. Boehm, "Get Ready for Agile Methods, with Care", IEEE Computer, Vol.35, No.1, 2002, pp.64-69.

[9] L. Wang, "Agility counts in developing small-size software", IEEE Potentials, Vol.26, No.6, 2007, pp.16-23.

[10] M. Pikkarainen, *Agile Assessment Approach (based on the eight case experiences)*, 2006, VTT publication.
[11] B. Boehm, R. Turner, "Using Risk to Balance Agile and Plan-Driven Methods", IEEE Computer, Vol.36, No.6, 2003, pp.57-66.
[12] D. Cohen, M. Lindvall, P. Costa,. "An introduction to agile methods", Advances in Computers, Vol.62, 2004, pp.1-66.
[13] C. Schwaber, *Enterprise Agile Adoption in 2007*, 2008, Forrester Research.
[14] O. Salo, P. Abrahamsson, "Agile Methods in European Embedded Development Organizations: a survey study of Extreme Programming and Scrum", IET Software, Vol.2, 2008, pp.58-64.
[15] T. Dyba˚, T. Dingsøyr, "Empirical studies of agile software development: A systematic review, Information and Software Technology", Vol.50, No.9-10, 2008, pp.833-859.
[16] A. Begel, N. Nagappan, *"Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study", Proceedings of the 1st International Symposium on Empirical Software Engineering and Metrics*, 2007.
[17] B. Fitzgerald, G. Hartnett, K. Conboy, "Customizing agile methods to software practices at intel Shannon", European Journal of Information Systems, Vol.15, No.2, 2006, pp.200-213.
[18] J. Sutherland, K. Schwaber, *The scrum papers: nuts, bolts, and origins of an agile method*, Scrum Inc, 2007.
[19] D. West, T. Grant, *Agile Development: Mainstream Adoption Has Changed Agility*, 2010, Forrester Research.
[20] J. Highsmith, *Agile software development ecosystems*, Addison-Wesley, Boston, USA, 2002.
[21] J. Sutherland, G. Schoonheim, E. Rustenburg, *"M. Rijk, Fully Distributed Scrum: The Secret Sauce for Hyperproductive Offshored Development Teams", Proceedings of the Agile Conference 2008*, Toronto, Canada, 2008, pp.339-344.
[22] M. Cohn, *User Stories Applied: For Agile Software Development*, Addison-Wesley, Boston, MA, 2004.
[23] G. Lee, W. DeLone, J.A. Espinosa, "Ambidextrous coping strategies in globally distributed software development projects", Communications of the ACM, Vol.49, No.10, 2006, pp.35-40.
[24] P. Ågerfalk, B. Fitzgerald, "Flexible and distributed software processes: old petunias in new bowls", Communications of the ACM, Vol.49, No.10, 2006, pp.27-34.
[25] P. Abrahamsson, O. Salo., J. Ronkainen, J. Warsta, *Agile software development methods: review and analysis*, 2002, VTT Technical report.
[26] A. Qumer, B. Henderson-Sellers, "An evaluation of the degree of agility in six agile methods and its applicability for method engineering", Information and Software Technology, Vol.50, 2008a, pp.280-295.
[27] A. Qumer, B. Henderson-Sellers, "A framework to support the evaluation, adoption and improvement of agile methods in practice", The Journal of Systems and Software, Vol.81, 2008b, pp.1899-1919.
[28] T. Chow, D. Cao, "A survey study of critical success factors in agile software projects", The Journal of Systems and Software, Vol.81, 2008, pp.961-971.
[29] A. Sidky, J.D. Arthur, S.A. Bohner, "A disciplined approach to adopting agile practices: the agile adoption framework", Innovations in Systems and Software Engineering, Vol.3, No.3, 2007, pp.203-216.
[30] G. Benefield, *"Rolling out agile in a large enterprise", Proceedings of the Hawaii International Conference on Software Systems (HICSS'41)*, 2008, pp.461-470.
[31] R.T. Johannesen, *An Investigation into Software Process Improvement in the Small and its Application in a Scandinavian Picture Agency Group*, 2004, University of Sunderland School of Computing and Technology.
[32] S. Lee, H.-S. Yong, "Distributed Agile: Project Management in a Global Environment", Empirical Software Engineering, Vol.15, No.2, 2010, pp.204-217.
[33] A. Cockburn, *Agile Software Development*, Addison-Wesley, Boston, MA, 2002.

Seiyoung (Sarah) Lee

She is a Deputy Director of Software Engineering Center at National IT Industry Promotion Agency, Republic of Korea. She has a Ph.D. degree in Computer Science and Engineering at Ewha Womans University and over 15 years of industrial experience in South Korea and California. She is a software engineering professional with the majority of her career involved in development, management and deployment of IT products worldwide. Most recently, she worked at Yahoo! Inc. as Senior International Program Manager and led international deployment projects for My Yahoo! and Front Page. Her main research interests are in Agile Methods, Software Process Improvement, Product Line Engineering and Software Globalization

Hwan-Seung Yong

He is a Professor of Computer Science and Engineering at Ewha Womans University, Republic of Korea since 1995. He received the B.S., M.S. and Ph.D. degrees in Computer Engineering from Seoul National University. He has five years of industrial experience as Researcher at ETRI (Electronics and Telecommunications Research Institute) and served as a visiting scientist at the IBM T.J. Watson Research Center. He is member of KIISE (The Korean Institute of Information Scientists and Engineers) and ACM. His current major research interests include agile methods, data mining, multimedia database and ontology search