International Journal of Software Engineering and Its Applications
Vol.8, No.5 (2014), pp.85-114
http://dx.doi.org/10.14257/ijseia.2014.8.5.09

An Exploratory Study on Agile based Software Development Practices

ShafinahFarvinPackeerMohamed[1], FauziahBaharom[1]and Aziz Deraman[2]

[1]*School of Computing, Universiti Utara Malaysia*
[2]*School of Informatics and Applied Mathematics Universiti Malaysia Terengganu*
shafinah, fauziah@uum.edu.my, a.d@umt.edu.my

Abstract

Agile based software development approach is increasingly gaining much interest among software practitioners and researchers due to its ability to produce high quality software in shorter period of time. Even though its importance has been revealed, only few studies were conducted regarding its current practices in the software industry, particularly in Malaysia. Thus, an exploratory study was conducted among software practitioners in Malaysia to study their practices and perceptions on the agile based software development in the real-world projects. Moreover, the practices were covered in a wider perspective by considering the software development, management and documentation practices. This paper discusses the findings from the study, which involved 73 agile practitioners. Structured questionnaire was used for data collection purpose while statistical methods such as frequency, mean, and cross tabulation were used for data analysis. Outcomes from this study reveal that the most implemented agile principles are early delivery and face-to-face communication, whilst the least implemented is self-organizing team. Additionally, majority of the respondents agreed that agile based software development practices are important in order to produce high quality software.

Keywords: *Agile based software development practices, exploratory study, software practitioners in Malaysia*

1. Introduction

Agile based software development approach provides ways to develop software faster based on a set of basic principles: 1) individuals and interactions over processes and tools; 2) working software over comprehensive documentation; 3) customer collaboration over contract negotiation; and 4) responding to change over following a plan [1]. It has emerged due to the problems faced in conventional methodologies which are not flexible in accepting unstable and volatile requirements. This is because current business environment emphasizes on faster delivery, low cost and ability to move and change quickly. At the same time, the produced software is expected to be high in quality. Consequently, nowadays software developers need to incorporate agility during software development process to fulfill these needs, as it is aimed to produce higher quality software in a shorter period of time [2,3]. Despite the importance of incorporating agility during software development, only few studies related to the current industrial practice of agile based software development practices have been conducted in Southeast Asia region, particularly Malaysia. Most of the other studies were conducted in Western countries [4]. Moreover, these studies only focus on the practices of a particular agile method.

ISSN: 1738-9984 IJSEIA
Copyright ⓒ 2014 SERSC

Based on the above limitations, an exploratory study was conducted to investigate the software practitioners' practices and perceptions on the software development practices concerning the agile based software development, by considering the software development, management and documentation practices. The research questions tried to be answered by this study are as follows:

1. How far is the software practitioners' experience in agile based software development approach?
2. How far do the software practitioners follow the agile principles?
3. What are the agile based software development practices that are important towards producing high quality software?
4. What are the important characteristics that a team and organization should possess towards successful implementation of agile based software development?

The following sections are organized as follows: Section 2 provides an overview of the agile methods and principles, then continued with Section 3 which provides the existing empirical studies on agile. In Section 4, the research approach is presented, followed by the findings in Section 5. Section 6 presents the discussions, continued with the limitations and implications of the study in Section 7. This paper ends with the conclusions in Section 8.

2. Agile Methods and Principles

Currently, there are various agile methods which include Extreme Programming (XP), Scrum,Adaptive Software Development (ASD), Crystal Methodologies, Dynamic Systems Development Method (DSDM), Feature-Driven Development (FDD), Lean Software Development and Agile Modeling (AM).Each of these methods place emphasis on certain phases and practices in the software development lifecycle.Some focus more on the software development practices, such as XP and AM, while some on the management of the software development practices, such as Scrum. The DSDM fully supports the software development lifecycle and the FDD is more suitable for the requirement specification phase [5]. Nevertheless, these methods have similar values and practices [6], whereby they follow the 12 principles aligned in the Agile Manifesto as listed in Table 1. Moreover, these agile methods also follow the agilefour values which are individuals and interactions, working software, customer collaboration and responding to change [1].

Table 1. Agile Principles [1]

No.	Principles
1	Satisfy the customer through early and continuous delivery of valuable software
2	Emphasize on face-to-face conversation for conveying information to and within a development team,
3	Emphasize on simplicity throughout the development process (estimation, design, etc)
4	At regular intervals, the team reflects on how to become more effective in future iterations/sprints,
5	Continuous attention is given to technical excellence and good design,
6	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
7	Working software is the primary measure of progress
8	The sponsors, developers, and customers maintain a sustainable development
9	The projects are built around motivated individuals

 Copyright ⓒ 2014 SERSC

10	Customers work closely with the agile team and are readily available
11	Welcome changing requirements, even late in development
12	Self-organized teams (team members make their decisions and plans without depending on managers)

In this study, the core practices of the agile based software development were derived from XP, Scrum and AM, as well as the agile principles and values [1]. These methods were chosen since they are the most popular [5]. In addition, XP and Scrum complement each other, whereby Scrum focuses on project management and XP focuses on project development [7]. The AM was also referred, as it is claimed to provide a methodology for effective agile modeling and documentation [8]. By taking these methods into consideration, the practices are covered from a wider perspective instead of only focusing on the software development practices.

3. Existing Empirical Studies on Agile based Software Development

There are many studies conducted in the software development area, intended for investigating the current practices of software development in Malaysian software industry such as [9-11]. However, only few related studies of the agile based software development practices in the industry have been conducted in Southeast Asia region, particularly Malaysia [12-14]. Most of the other studies were conducted in Western countries [15-53]. Dyba and Dingsoyr [2] have systematically reviewed the empirical studies related to agile that are available in the literature up to year 2005.

The survey on the state of agile adoption was started by VersionOne [15] since 2006 with the aim of getting the insight on the status of the agile development adoption and practices. In 2011, 6042 respondents have participated in the survey. Two indications found from the survey are (i) more than 80% of the respondents were practicing agile, and (ii) the most used agile methods were Scrum and Scrum/XP hybrid. Questions related to the agile based software development practices were also included in the survey. The survey concluded that there was an increased use in Kanban.

Forrester Research conducted a similar survey participated by 1298 respondents of application development and program management professionals, who were more familiar with Scrum [16]. The results showed that among the most used agile practices were short iterations (79%), constant feedback (77%) and daily scrum meeting (71%), while the least used were test driven development (TDD) (42%) and metaphor (15%). On the other hand, Salo&Abrahamsson in [17], who have investigated the usefulness of XP and Scrum in the European embedded software development organizations, found out that open office workspace (66%), coding standards (60%) and 40-hour week (59%) to be the most used agile practices. The least practiced were TDD and pair programming. In terms of the Scrum practices, product backlog and daily scrum meeting were highly considered.

Through an online survey which was participated by respondents from various countries (*e.g.*, Brazil, United States), Santos *et al.*, [18] studied the perception of software practitioners on the relationship of agile practices with the quality of software. The findings revealed that with a bigger number of staff involvements, agile management of the proposed requirements and code development can lead to a higher software quality. While conducting two case studies to explore the implementation of agile practices among software practitioners in Philippines, Sison& Yang [26] indicated that one of the participated organizations implemented only five of XP practices (coding standards, small releases, simple design, metaphor, and 40-hour week). Another organization that implemented Scrum appreciated the

Scrum's sprint and its sub practices, as well as the daily Scrum meetings because it helped in improving productivity.

As for Malaysia, Ani Liza and her colleagues [12] investigated on the perception of software practitioners when adopting agile. In their study, they concluded that agile was still new and emerging method in Malaysia and the awareness was still low especially within the government sector. In another study, Ani Liza and her colleagues [13] investigated on the issues and problems faced by the early agile methods adopters. In their study, they concluded that social and human were the most important factors in using the agile method, while the technical factor was less important. On the other hand, Mazni, Sharifah-Lailee and Azman [14] studied on the impact of agile approach among the software engineering teams in a computer center in Malaysia. However, in this study, it was found that effective methodology and organizational culture were two important factors that must be considered in producing innovative teams and quality software. Table 2 summarizes the main findings of the existing studies.

Table 2. Summary on Main Findings of Existing Studies

Issues	Descriptions	Authors
Most used agile methods	Scrum and Scrum Hybrid	[15]
	Scrum	[16]
	XP	[17]
Most used agile practices	Short iterations, constant feedback, product owner, daily Scrum meetings	[16]
	Open office workspace, coding standards, and 40-hour week	[17]
	Product backlog and daily Scrum meeting	[17]
	Coding standards, small releases, simple design, metaphor and 40-hour week	[26]
	Sprint and its sub practices, daily Scrum meeting	[26]
	The involvement from all parties from the beginning, daily stand-up meeting, iterative and incremental, applying burn down chart, sprint and continuous integration	[12]
Least used agile practices	Test-driven development, system metaphor	[16]
	TDD and pair programming	[17]
Benefits of agile	XP increased good relationship among developers	[26]
	Scrum usage has improved their productivity	[26]
Factors that influence the successful implementation of agile	Change in mind set of the people in organization	[12]
	Social and human factors	[13]
	Effective methodology and organizational culture	[14]
	Bigger involvement of the staff, agile management of the requirements proposed and code developed	[18]

 Copyright ⓒ 2014 SERSC

Based on the existing studies discussed, only few studies were conducted among Malaysian software practitioners regarding agile practices, whereas most of the studies were performed in the Western countries. In addition, their focus was more on a particular agile method, such as Scrum or XP, or combination of Scrum and XP. On top of that, the research on the effectiveness of AM practices is stil scarce [54]. Therefore, in this study, the practices of agile are considered from wider perspectives, whereby the practices of XP, Scrum and AM are gathered and mapped to the phases of software development. This is intended to cover the overall process of software development, including management and documentation. Additionally, the importance of the AM practices among software practitioners can be revealed. Section 4 explains about the execution of the study.

4. Research Approach

The study was conducted by using structured questionnaire because of the cost effectiveness, easiness in doing analysis, wider area coverage and integrity assurance [77]. There were five main activities involved, which started with instrument design, sampling, pilot study, data collection and data analysis [9]. The instrument was constructed by referring to the previous works such as [9, 15, 18, 25, 47, 55]. It consisted of 29 questions with sub questions, organized into two main sections: demographic background and agile based software development practices. In general, 7-point semantic differential scale ranging from Unimportant to Very Important was used for most of the questions. Additionally, multiple responses questions and yes/no questions were also included [56].

In this study, the purposive sampling was used, which involved the selection of unique sample with specific feature that is important for the study [56]. The sample for this study was chosen among the software developers in Malaysia. Prior to the real study, a pilot study has been conducted to confirm the validity and readability of the questionnaire. According to Teijlingen and Hundley [57], pilot test is very important in investigating the wellformedness and feasibility of the questionnaire. Thirty two (32) respondents were chosen to answer and give feedback about the instrument. The respondents include system analysts and programmers who have at least 3 years' experience. They gave some suggestions for improving the quality of instrument, including simplifying the questions to be more readable and understandable, reducing the number of questions and reorganizing the presentation of questions. Consequently, the instrument was refined based on their feedback.

To conduct the real study, the researcher contacted the potential respondents through telephone or e-mail and asked their willingness to participate in the study. The ones who were willing to participate in the study answered the questionnaire either through online survey, or email or by hardcopy (sent by post to them). To ensure the validity of the collected data, the respondents were only chosen among the ones who had basic knowledge in the agile based software development process. Most of the respondents were from Kedah, Penang, Kuala Lumpur and Selangor, as these are the places where software development companies are concentrated in Malaysia [13]. Moreover, these states have the big software technology parks and International software development organizations. Data obtained from the exploratory study were analyzed using descriptive statistical analysis. The analysis is not intended to explain or show causal relationships between the variables, rather it focuses on describing what proportion of a sample has a certain opinion or how often certain events occur [58]. Among the analysis used were frequency, mean and cross tabulation. The SPSS software was used for this purpose.

Copyright ⓒ 2014 SERSC

5. The Findings

This section discusses the results obtained from the study. They are presented based on the items in the questionnaire and the research questions of the study.

5.1. Demographic Data

The respondents were asked about their position and experience in software development. Cross tabulation analysis is used to classify them, as depicted in Table 3. Most of the respondents were programmers (43.8%). Out of the 73 respondents, only 12 (16.4%) had experience more than 10 years, while majority of them had 1 to 5 years experience (52.1%).

Table 3.Respondents' Experience

Positions	**<1 year**	**1-5 years**	**6-10 years**	**11-20 years**	**Total**
Programmers	7 (9.6%)	21 (28.8%)	2 (2.7%)	2 (2.7%)	32 (43.8%)
System Analysts	1 (1.4%)	8 (11%)	7 (9.6%)	2 (2.7%)	18 (24.7%)
Project Managers	1 (1.4%)	3 (4.1%)	2 (2.7%)	4 (5.5%)	10 (13.7%)
Quality Assurance/Testers	0 (0%)	5 (6.8%)	0 (0%)	1 (1.4%)	6 (8.2%)
Team Leaders	0 (0%)	1 (1.4%)	1 (1.4%)	3 (4.1%)	5 (6.8%)
Security Advisors	2 (2.7%)	0 (0%)	0 (0%)	0 (0%)	2 (2.7%)
Total	**11 (15.1%)**	**38 (52.1%)**	**12 (16.4%)**	**12 (16.4%)**	**73 (100%)**

The respondents worked in software development, education/training, service and public administration, manufacturing, telecommunication, consultation, banking/financial/insurance or health and social work sector. The information is presented in Table 4. Most of the respondents were from private sectors (85%), with 52.1% of them from software development organizations.

Table 4. Classification of Organization Sector

Sectors	**Organization Types**		**Total**
	Private	**Government**	
Software Development	38 (52.1%)	0 (0%)	38 (52.1%)
Education/Training	7 (9.6%)	8 (10.9%)	15 (20.5%)
Service and Public Administration	5 (6.8%)	1 (1.4%)	6 (8.2%)
Manufacturing	4 (5.5%)	0 (0%)	4 (5.5%)
Telecommunication	4 (5.5%)	0 (0%)	4 (5.5%)
Consultation	3 (4.1%)	0 (0%)	3 (4.1%)
Banking/Financial/Insurance	1 (1.4%)	0 (0%)	1 (1.4%)
Health & Social Work	0 (0%)	2 (2.7%)	2 (2.7%)
Agriculture, Hunting & Forestry	0 (0%)	0 (0%)	0 (0%)
Total	**62 (85%)**	**11 (15%)**	**73 (100%)**

 Copyright ⓒ 2014 SERSC

In terms of the size of their companies (Figure 1), 29% of them worked in large companies, whereby their organizations had more than 250 employees. Similarly 29% of them worked in small companies which had less than 20 employees.

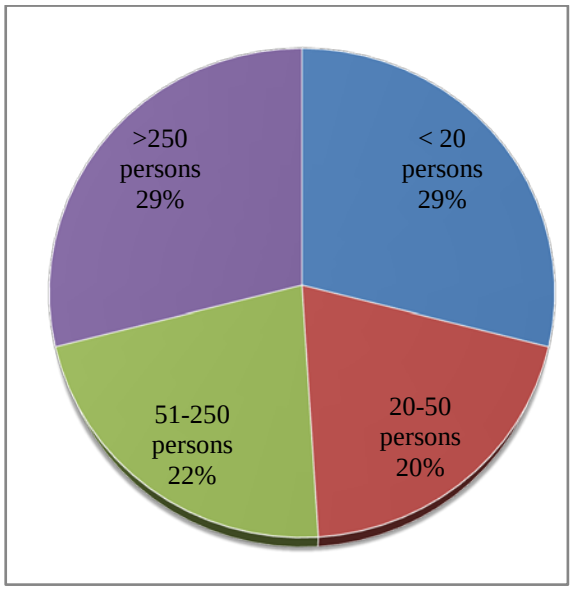

Figure 1. Number of Employees in Organization

5.2. Software Practitioners' Experience and Perceptions on Agile based Software Development

This section presents the results based on the stated research questions.

Research Question 1: How far is the software practitioners' experience in agile based software development approach?

- Familiarity with the agile based software development approach

When asked about their experience in implementing the agile based software development approach, 25% of them mentioned that they have heard about agile but not in depth, while the rest (75%) were either current member of agile (30%) or currently leading agile team (19%) or previously were in agile team (16%) or agile coach (10%). Figure 2 shows the related analysis result.

Copyright ⓒ 2014 SERSC

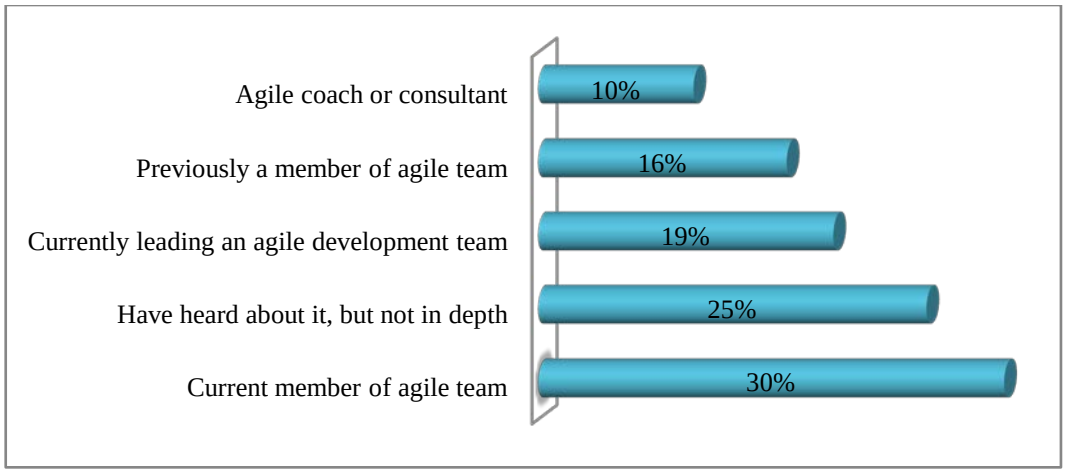

Figure 2. Level of Exposure for Agile based Software Development Approach

They were then asked about their years of experience in the agile based software development. Majority of them (78%) had experience in the agile based software development for the period of two years or less, while only 7% had experience in it for five years or more (Refer to Figure 3).

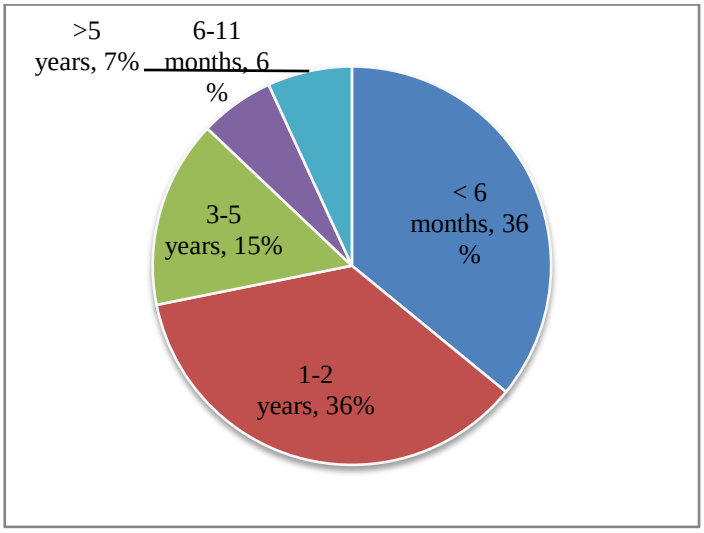

Figure 3.Exposure on Agile based Software Development

- The number of team members

Figure 4 depicts the number of team members in the respective respondents' team. Most of them worked in a team with less than five members (37%) or five to ten members (34%).

Copyright ⓒ 2014 SERSC

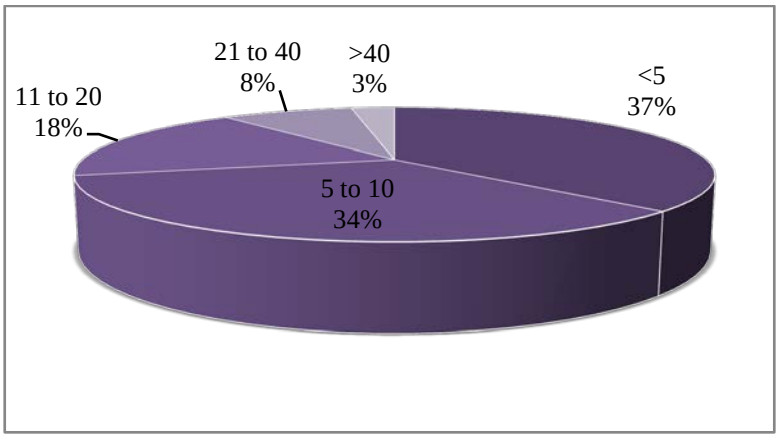

Figure 4.Number of Agile Team Members

- The agile methods being practiced

Figure 5 presents the agile methods that the respondents were familiar with. They were allowed to choose more than one answer for this question. Most of them were familiar with Extreme Programming (XP) (52%), followed by Scrum (32%).

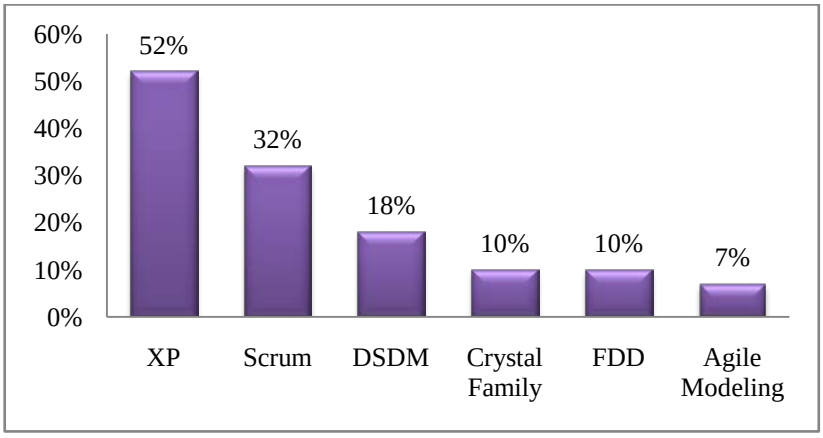

Figure 5. Agile Methods Being Practiced

- Opinion on the importance of incorporating agility in software development to ensure the quality of software

Regarding their agreement that agility can influence the quality of producing software, 96% of the respondents answered 'Yes'. This is in line with the literature, whereby it is widely discussed and accepted that the agile based software development is essential and should be included in the current software process as high quality software could be marketed faster in the most cost-effective manner [60,61]. Additionally, the respondents were also asked about the benefits that they gained by practicing agility. This question allowed multiple answers. The results indicate that most of them agreed that agility can enhance the ability to manage changing requirements (59%), and increase productivity (55%). Figure 6 shows the analysis result.

Copyright ⓒ 2014 SERSC

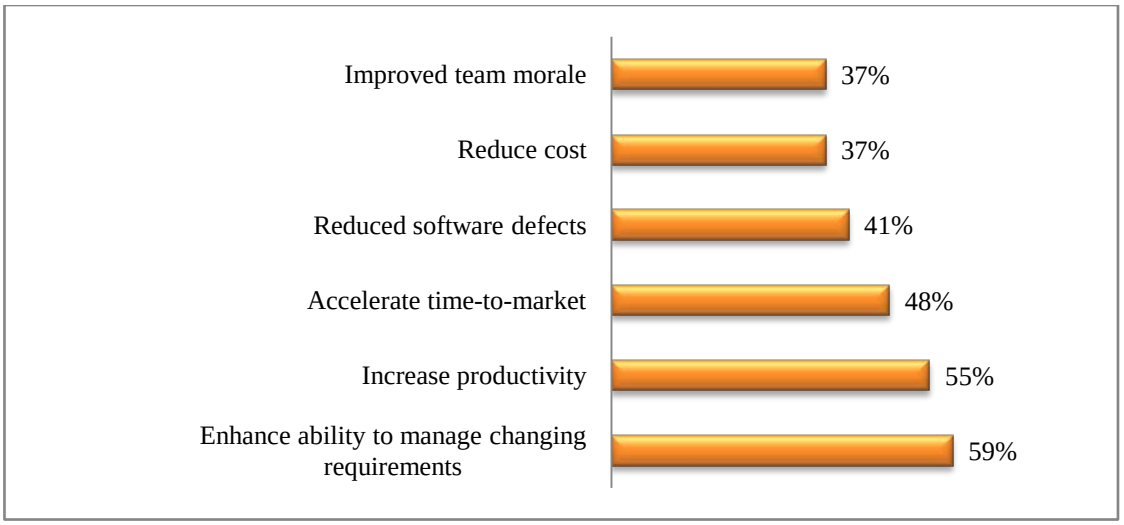

Figure 6. Benefits of Agility Practices

Research Question 2: How far do the software practitioners follow the agile principles?

The respondents were then asked about the frequency of the agile principles being practiced in their organizations. The 7-point numerical scale [62] was used for this question, which ranged from Never to Every time. This scale was then mapped to equal intervals, as depicted in Table 5. The results in Table 6 demonstrate that none of the agile principles was used 'Every time' by the respondents. Nevertheless, majority of the principles were performed frequently. Only principles 1 to 5 were performed 'Usually'. This may indicate that the software practitioners only claimed that they have implemented agile based software development; however they did not follow the principles all the time.

Table 5.Interval Values

Degree of importance	Interval value
Never	1.00 – 1.86
Rarely	1.87 – 2.72
Occasionally	2.73 – 3.58
Sometimes	3.59 – 4.44
Frequently	4.45 - 5.30
Usually	5.31 - 6.16
Every time	6.17 - 7.00

Table 6. Agile Principles Implementation

Agile principles	Mean	DOF
1) Satisfy the customer through early and continuous delivery of valuable software	5.82	Usually
2) Emphasize on face-to-face conversation for conveying information to and within a development team	5.67	

Copyright ⓒ 2014 SERSC

3) Emphasize on simplicity throughout the development process (estimation, design, coding, etc)	5.34	
4) At regular intervals, the team reflects on how to become more effective in future iterations/sprints	5.33	
5) Continuous attention is given to technical excellence and good design	5.32	
6) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale	5.30	Frequently
7) Working software is the primary measure of progress	5.14	
8) The sponsors, developers, and customers maintain a sustainable development	5.12	
9) The projects are built around motivated individuals	5.08	
10) Customers work closely with the agile team and are readily available	5.01	
11) Welcome changing requirements, even late in development	4.88	
12) Self-organized teams (team members make their decisions and plans without depending on managers)	4.45	

Research Question 3: What are the agile based software development practices that are important towards producing high quality software?

The respondents were further inquired about the software development practices that need to be performed in order to produce high quality software, concerning the agility practices. The practices were categorized into requirement engineering, design, coding, testing, project management and change management. The mean value for each practice is used in the analysis, as it represents the most selected answers by respondents. The 7-point numerical scale was used for this question, which ranged from Unimportant to Very Important. The scale was then mapped to equal intervals, as depicted in Table 7. This is followed by the mean values obtained by the important practices in each phase of the agile based software development in Table 8. Outcomes from the study show that mostly these practices obtained high consideration among the respondents, whereby the mean values are in the range of Important to Very Important. This shows that they are important practices in producing high quality software. In addition, it indicates that the opinion of the respondents is aligned with the literature.

Table 7.Interval Values

Degree of importance (DOF)	Interval value
Extremely Not Important (ENI)	1.00 – 1.86
Not important (NI)	1.87 – 2.72
Less Important (LI)	2.73 – 3.58
Moderately Important (MI)	3.59 – 4.44
Important (I)	4.45 - 5.30
Very Important (VI)	5.31 - 6.16
Extremely Important (EI)	6.17 - 7.00

Copyright ⓒ 2014 SERSC

Table 8. Agile based Software Development Practices

Phases	Practices	Mean	DOF
Requirement Engineering	1. Identifying the scope at the beginning of the project to create initial prioritized stack of requirements	5.58	(VI)
	2. Gathering requirements iteratively and incrementally	5.51	
	3. Emphasizing on face-to-face communication	5.51	
	4. Producing product backlog and iteration backlog for ensuring the consistency and traceability of requirements	5.42	
	5. Emphasizing on single source information	5.30	(I)
	6. Using releases (working software) for validating requirements at the end of each iterations	5.23	
	7. The requirements are written on cards in short statements	5.11	
	8. Enabling development team to re-estimate the time and velocity (speed of accomplishing tasks) of user stories	5.03	
	9. Enabling customers to prioritize and reprioritize requirements throughout the development	4.42	(MI)
Design	1. Implementing model storming	5.36	(VI)
	2. Creating an initial model at the beginning of iteration	5.27	(I)
	3. Start designing with simple initial design and integrating it continuously	5.21	
	4. Producing just barely good enough artifacts (for situation at hand only)	5.07	
	5. Refactoring (reorganize) the design	4.85	
	6. Using metaphor as architecture of the system	4.64	
Coding	1. Delivering the features with high priority first	5.59	(VI)
	2. Following coding/database/interface standards	5.47	
	3. Delivering the software frequently with increments of features	5.42	
	4. Deploying the software gradually in real environment	5.42	

Copyright ⓒ 2014 SERSC

Phases	Practices	Mean	DOF
	5. Having customers on-site to get continuous and immediate feedback from customer for clarification	5.34	
	6. Integrating the newly produced code to system baseline frequently	5.14	(I)
	7. Determining code integration strategy and revising it	5.05	
	8. Implementing test driven development (TDD): write tests first, then write the code to pass the tests	5.01	
	9. Producing deliverable documentation late	5.01	
	10. Refactoring the code and database	4.85	
	11. Implementing pair programming (two programmers working together)	4.79	
	12. Giving authority to team members to make changes at any part of the code	4.77	
Testing	1. Implementing user interface testing	5.77	(VI)
	2. Using acceptance tests to validate and verify user's requirements	5.64	
	3. Implementing database regression testing	5.51	
	4. Producing executable specification	5.41	
	5. Acceptance tests are written or at least modeled by customers	5.26	(I)
	6. Implementing automated tests	5.23	
	7. Implementing frequent integration testing	5.23	
	8. Implementing tests continuously throughout the development	5.18	
Project Management	1. Performing project planning jointly and continuously with team members	5.41	(VI)
	2. Conducting continuous review meetings at end of each iteration to demonstrate the latest version of software	5.32	
	3. Planning and estimating (cost and schedule) are based on features/ functions/stories of system	5.23	(I)

Copyright ⓒ 2014 SERSC

Phases	Practices	Mean	DOF
	4. Revealing the current progress of iteration/sprint everyone on sprint burn down chart	5.19	
	5. Carrying out release meeting at the beginning of project to plan releases	5.18	
	6. Ensuring that working hours do not exceed 40 hour per week (no overtime)	5.14	
	7. Carrying out iteration meeting at the beginning of each iteration to plan iterations	5.11	
	8. Conducting retrospective (postmortem) at end of each iteration to look back what worked well and what need to be improved	4.81	
	9. Monitoring customer involvement and end-user in project activity	4.71	
	10. Carrying out daily stand-up meetings for daily plan	4.68	
Change Management	1. Controlling changes using product backlog (prioritized user stories)	5.40	(VI)
	2. Assigning the individual who will be responsible for ensuring Change Management activities are implemented correctly	5.12	(I)
	3. Automating the Change Management activities (e.g. building scripts)	4.99	
	4. Not allowing changes once an iteration has begin, until the iteration ends	4.93	

Research Question 4: What are the important characteristics that a team and organization should possess towards successful implementation of agile based software development?

In addition, the respondents are asked about the team and organizations characteristics that are important in order to successfully implement the agile based software development. The 7-point numerical scale is used for this question and the interval representation used as in Table 8. Table 9 shows the team and organizations characteristics and their mean values.

Table 9. Team and Organizations Characteristics

Types	Characteristics	Mean	DOF
a	1. Emphasize on face-to-face communication	5.97	(V)

 Copyright ⓒ 2014 SERSC

Types	Characteristics	Mean	DOF
	2. The team members consists of people with different functional expertise	5.96	
	3. Small sized team	5.60	
	4. Co-located team	5.23	(I)
	5. Self-organized team	5.14	
Organization	1. Encourage customer participation	5.96	(VI)
	2. Provide cooperative organizational culture instead of hierarchical	5.86	
	3. Provide facilities with proper agile-style work environment	5.73	
	4. Ensure that agile way of software development is universally accepted	5.71	

6. Discussions

As a whole, this study has answered all of the stated questions. Findings from the study found that majority of the respondents (75%) have had experience in the agile based software development. This indicates that the agile based software development is increasingly being implemented by the software practitioners. However, there were still among them who only briefly heard about it. Furthermore, majority of them worked in small teams, as encouraged by Scrum, whereby the team members were less than ten (10). This practice is essential for agile as it involves frequent communication. Having more people will make communication harder [20, 49]. Furthermore, the result highlights that agile is being implemented not only in small companies, but also in large companies, as well as small-medium enterprise companies. The most used agile methods were XP and Scrum, akin to the results obtained in the previous studies [5, 15, 16, 17, 59]. One of the main reasons is that XP and Scrum complement each other, since Scrum focuses on project management, while XP focuses on project development [7]. AM was the least implemented method, although it provides effective way of modeling and documenting in the agile based software development, as claimed by Ambler [8]. Moreover, majority (96%) of the respondents agreed that agility should be considered during software development in order to produce high quality software. Among the benefits gained are enhancing the ability to manage changing requirements, enhancing software quality and accelerating time-to-market.

Next is about the agile principles implemented. The most implemented agile principles (achieved ‘Usually’) by the respondents were satisfying customers through early and continuous delivery of software, emphasizing on face-to-face conversation, emphasizing on simplicity throughout the development process, enabling the team to reflect on how to become more effective and giving continuous attention to technical excellence and good design. While the least implemented was self-organizing team. The respondents performed the agile practices according to the agile principles, however not constantly. These principles are essential to be considered by the software practitioners as they are the backbone of the

Copyright ⓒ 2014 SERSC

agile based software development process. If they are violated, it means that the agile based software development process is not being implemented properly. Furthermore, this might cause the agile values are not delivered properly to the customers [79].

Then the discussion is continued with the agile based software development practices. The agile based software development practices and the characteristics of the agile team and organization included in the questionnaire have been agreed by the software practitioners as essential practices that need to be performed towards producing high quality software. As mentioned before, basically these practices were derived from XP, Scrum and AM, as well as the agile principles and values in the Agile Manifesto. They are discussed below and summarized in Table 10.

- **Agile Requirement Engineering**

Following the agile principles, agile requirement engineering is performed iteratively and incrementally, in contrast with the conventional software development approach which emphasizes completed and well-defined requirements up-front [22, 36, 38, 44]. In this way, the requirements evolve over time throughout the development. Furthermore, agile give importance on face-to-face communication during the requirement elicitation, with minimal documentation. These practices are identified as very important practices by the respondents of this study that aligned with the previous studies by [19, 21, 44], as well as [22]. Furthermore, towards ensuring the consistency and traceability of requirements, the uses of product and iteration backlog have been agreed by the respondents as 'very important'. Similar result is reported in [17]. Additionally, the scope is identified at the beginning of the project to create initial prioritized stack of requirements, as emphasized by Ambler [8] and performed by O'Sheedy and Sankaran [23] in their studies.

Moreover, the requirements are gathered with little detail in the beginning of the project and detailed up during iterations through discussions and negotiations [19]. Additionally, the developers are able to re-estimate the time and velocity of accomplishing the requirements [8, 63]. In order to verify the requirements and show the progress to customers after completing each iteration, the working software (releases) are demonstrated to the customers. These practices have been rated as important agile requirement engineering practices for ensuring software quality by the respondents of this study. Similar results were obtained in the studies of [21, 22, 44]. However, even though the studies of Liu, Wang and Gao[44], Ramesh, Lan and Baskerville [21], and Lan and Ramesh [22] found that the respondents appreciated when the customers continuously prioritized the requirements, the result of this study is contradict. The result obtained for this practice is 'moderately important'.

In addition, emphasizing on the single source information also reduces the maintenance and traceability burden, as well as increases the consistency [8]. The documentations produced are minimized by documenting repeating information only once, such as the business rules. This single source of information can be a reference in producing other documents, rather than repeating them again and again [8]. However, the importance of this practice has not been studied previously.

- **Agile Software Design**

The agile based software development approach emphasizes on simple initial design which continuously evolve over time, in contrast with the traditional approach which design everything up-front. Simple design is one of the agile based software development's success factors concluded by Rumpe and Schröder, as well as Tsun and Dac-Buu, Sison and Yang and Tessem in their studies [24, 25,26,27]. Similarly, this practice was rated as 'important' in

 Copyright ⓒ 2014 SERSC

this study. Designing in simple way can be accomplished by producing just barely good enough artifacts/documents. This means to produce documentation for the situation in-hand only, rather than documenting the whole project. This is done by modeling and documenting during the iterations. The iteration modeling is implemented during each of the iteration planning meetings, whereby the requirements selected to be implemented in the particular iteration are modeled. The detailed modeling is then implemented through model storming for the in-hand solution before the development. The issues that need to be resolved is identified and explored together in a small group. During this discussion, the models are sketched on the whiteboard or paper and made visible to everybody [8]. These practices have not been studied by previous studies. However, in this study, model storming attained 'very important', while iteration modeling and producing just barely good enough artifacts obtained 'important'.

Furthermore, refactoring is another important agile design practice. It is a valuable tool that can be used to improve the software design [25, 28, 29]. This is also agreed by the respondents in this study. Besides, the importance of metaphor was revealed by the respondents in Begel&Nagappan [30], as well as the result of this study. Conversely, studies by Rumpe and Schröder[24] and West and Grant [16] did not support this observation since this practice was least used.

- **Agile Coding**

Similar to the requirement engineering and designing, coding in agile is also implemented iteratively and incrementally. In addition, before starting the coding, all programmers have to agree upon a set of coding/database/interface standard that everybody will follow during development. This practice assists in giving better understanding on the code, improves communication and facilitates maintenance. Studies in [15, 17, 19, 24, 25, 26, 30], indicated that coding standard is a highly adopted practice among their respondents.

The next practice is about delivering softwarefrequently with features increments. By doing so, the software can be demonstrated earlier to customers and enable them to review the software, identify defects and make adjustment for future requirements [64]. This practice has been considered as very important by the respondents in this study, as well as in the studies by [24, 25, 26, 33]. This practice is closely related with deploying the software gradually in real environment, which gained high consideration in study by Williams and Erdogmus [31]. On contrary, in VersionOne's study [15] this practice was rated as low percentage.

In addition, having customer on-site facilitates in providing continuous and immediate feedback. It is one of the essential agile principles and has high influence on the success of agile [16, 25, 27, 37, 47, 65]. It denotes that customer should be a member in the development team so that the uncertainties can be cleared as soon as it occurs [64]. This is contradicting to the traditional software development approach, whereby the customers typically involved during the initial requirements elicitation. Only towards the end they will then give their feedback on the developed software [65]. Customer involvement has also been identified as an important practice in [12, 25, 27]. On contrary, study by Rumpe and Schroder [24] concluded that on-site customer was least implemented and hard to be performed.

Delivering features with high priority is one of the agile principles. It can ensure that most of the important business values are to be delivered first [39]. This agile principle is highly considered among the respondents in this study. Moreover, study by [33] concludes that it is one of the factors that influence the success of the agile implementation. Moreover, deploying the software gradually in the real environment reduces the risk of deploying all at once. Furthermore, the feedbacks can be obtained earlier [66]. All of the previously discussed coding practices obtained 'very important' for this study. The practices which obtained 'important' are discussed next.

Copyright ⓒ 2014 SERSC

Agile emphasizes collective code ownership, whereby all programmers in a team are empowered to make any changes to any part of the code they are working on. This practice gained high consideration among the respondents in this study. Even previous studies by [17, 19, 24] provide the same report. Moreover, pair programming is one of the most accepted and succeeded practices in the industry and academic. The most significant result is improvement in the quality of design and code, as reported by [30, 59, 67, 68]. This practice encourages two programmers to work together when accomplishing their tasks which enable them to transfer their knowledge as well as to review the code permanently [59]. Studies by [19, 24, 27, 30, 59, 68] found out that pair programming is among the beneficial agile practices. Conversely, studies by VersionOne and Salo&Abrahamsson [15, 17] found that this practice is rated as the least practiced. Although the respondents of Schindler [59] practiced pair programming, they did not use it regularly. They usually used it based on demand especially for complex code or debugging.

Test driven development (TDD) is a critical practice in producing high quality software [67]. The developers create the unit tests before writing the production code. Many studies have proven its ability to produce high quality software, such as Huang and Holcombe, Gupta and Jalote, and Desai and Janzen [69, 70, 71]. Meanwhile, Sanchez, Williams and Maximilien [72] stated that the complexity of code and design is reduced with this practice. Similarly, Desai and Janzen [71] reports the same in their study. Study by [19, 34, 35, 72, 78] signified that the TDD is an important practice, while studies by West and Grant, Salo and Abrahamsson, and Begel and Nagappan [16, 17, 30], gave contradict reports.

Continuous integration of source code to the system baseline has been found as an important practice in [15, 16, 17, 19, 24, 30]. By performing this practice, compatibility problems can be detected or avoided earlier [38]. In fact, the definition and revision of the code integration strategy has been included in the CMMI Version 1.3 [36], which indicates it as an important practice.

By practicing refactoring on code and database**,** the software will be easier to be understood, helps in finding bugs and performing program faster. The refactoring focuses on the internal code restructuring (attributes and methods) across existing classes, without changing its external behavior [29]. Moser and his co-researchers [28] pointed out that refactoring increases the software quality as well as improves the productivity. Additionally, this practice gained high consideration in study by Ambler [34]. On the other hand, Alshayeb [37] indicated that refactoring does not influence the quality (adaptability, maintainability, understandability, reusability, and testability) of the developed software. As for this study, refactoring is highly considered.

As agile involves frequent changes, the production of the deliverable documentations is deferred to the end of development. The documents are created just before delivering the software. In preparing documentation, this can be risky because the earlier details of the requirements or design might change [8]. However, the importance of this practice has not been studied in previous studies.

- **Agile Testing**

The agile testing practices attained either ‘very important’ or ‘important’, which indicates that the practices are important towards producing high quality software. Testing in the agile environment is done continuously throughout the development, as reported by [15] and [44]. It involves frequent unit, system integration, user interface, database regression, and user acceptance. This differs to the conventional approach which conducts testing after the implementation stage. Database regression testing attained high importance in the study of Ambler [34], while user interface tests was emphasized in studies by VersionOne [15] as well

 Copyright ⓒ 2014 SERSC

as [44]. Furthermore, the integration testing must be done frequently as performed by the respondents in [24, 25, 33].

Moreover, the user acceptance tests are written by the customers to assure that the systems fulfill their needs, as reported in [22, 39]. In cases where the customers do not have technical knowledge, the developers will help the customers in writing the acceptance tests. Referring to the study by Lan and Ramesh [22], the acceptance tests acts as a mechanism to validate and verify user's requirements. In addition, agile also emphasizes on automating these tests. This practice gained high consideration by the respondents in this study, as well as studies by [15, 19, 38, 44]. The well-written tests act as executable specification. For instance, unit test is a portion of technical documentation and acceptance test is part of requirement documentation [8]. However, the importance of this practice has not been studied for the real world implementation.

- **Agile Project Management**

Project management in agile is different than that in the conventional software development approach. The project management consists of three planning levels which obtained high consideration in previous studies; release plan, iteration plan [26] and daily plan [16, 17, 26, 40]. These planning are done iteratively and collaboratively, rather than planning the whole project up-front, as reported in [22, 44]. A study by Tessem [27] indicates that conducting these planning leads to better estimation of the work size. However, in Salo and Abrahamsson [17], the collaborative planning was rated as low. Additionally, the agile project management emphasizes on the sprint review and retrospectives which are held at the end of a sprint [22, 42]. The retrospective was found to be more beneficial when applied to small teams, participated by the whole team and when the comments are recorded [20]. Similarly, Sison and Yang concluded that retrospective is important [26]. Additionally, Sliger and Broderick [74] explained that the planning must be done according to the features/requirements. However, the importance of this practice has not been studied before.

In addition, the progress of the team should be revealed in an open space so that everyone is aware of the current progress of the project. This practice gained high consideration in this study, as well as in the studies by [15, 16, 19]. At the same time, the working hours should not exceed 40 hours in a week to ensure productivity. This practice obtained high consideration among the respondents not only in this study but also in Rumpe and Schroder, Salo&Abrahamsson, and Sison and Yang [17, 24, 26] studies. Furthermore, the involvement of customers and end-users is monitored by the management, as agile enforces their collaboration with the development team [36, 74].

- **Agile Change Management**

Since agile involves a lot of frequent changes, the changes need to be adapted, rather than controlling them. Thus, change management and traceability is imperative [75]. Furthermore, a particular individual who will be responsible in managing the changes must be identified [36]. To enable the change management activities to be more efficient, the change management activities are automated. For example, the use of automated tool for scripts creation. In order to avoid scope crepe, the changes are controlled by monitoring the product backlog and by restricting changes once the iteration starts [8, 42]. However, previous studies that studied about the importance of these practices are hardly found.

Copyright ⓒ 2014 SERSC

- **Team Characteristics**

It is widely accepted that software development processes is highly influenced by the team performances, as it involves human interactions. This is same with agile based software development approach, whereby emphasis is given on individuals and interactions, as well as customer collaboration and responds to changes, as opposed to the individual role assignment implemented in the traditional software development approach [65]. The respondents were asked about the characteristics that should exist among the team and the support that should be given by organization in order to successfully develop high quality software. The important factors that need to be considered in order to achieve high performance teams are through feedback and communication [76]. Therefore, agile stresses face-to-face communication and this practice was reported as beneficial in studies of [22, 39, 43, 44, 45]. In order to make these possible, the team must be small sized, as concluded by O'Sheddy and Sankaran [23] in their study. However, study by Misra and his colleagues reported as opposed [47]. Furthermore, they should be placed at the same workplace area. This is essential for intense interaction and knowledge sharing [49]. It is also agreed that team members must have high competence and expertise [25, 48, 49]. On contrary, collocated team was found as not beneficial in [47]. Furthermore, in agile, the self-organized team is essential because it can make decisions and plans without depending on managers [1, 25, 73, 74]. This practice has been identified as able to influence the team effectiveness [25, 33, 50].

- **Organization Characteristics**

Organization plays an important role in enabling the implementation of agile based principle on the team [13, 14, 25, 47, 53]. The organization must provide agile based environment throughout the organization by providing cooperative organizational culture instead of hierarchical, encouraging face-to-face communication, ensuring that the agile way of software development is universally accepted and offering facilities with proper agile-style work environment [25, 51]. In relation to the previous studies [13, 25, 49, 51, 52, 53] this study also indicates that these are the important practices.

Table 10.The Agile based Software Development Practices

Phases	Agile based Software Development Practices	Previous studies
Requirement Engineering	1. Gathering requirements iteratively and incrementally	[19, 21, 22, 44] (+)
	2. Emphasizing on face-to-face communication	[19, 21, 22, 44] (+)
	3. Identifying the scope at the beginning of the project to create initial prioritized stack of requirements	[8, 23](+)
	4. Producing product backlog and iteration backlog for ensuring the consistency and traceability of requirements	[17] (+)

 Copyright ⓒ 2014 SERSC

Phases	Agile based Software Development Practices	Previous studies
	5. Using releases (working software) for validating requirements at the end of each iterations	[21, 22, 44] (+)
	6. Enabling development team to re-estimate the time and velocity (speed of accomplishing tasks) of user stories	[21, 22, 44] (+)
	7. Emphasizing on single source information	[8](+)
	8. The requirements are gathered with little detail in the beginning and detailed up during iterations	[19](+)
	9. Enabling customers to prioritize and reprioritize requirements throughout the development	[21, 22, 44] (+)
Design	1. Implementing model storming	[8](+)
	2. Start designing with simple initial design and integrating it continuously	[24, 25, 26, 27](+)
	3. Refactoring (reorganize) the design	[25, 28, 29](+)
	4. Using metaphor as architecture of the system	[30] (+)
		[16, 24] (-)
	5. Creating an initial model at the beginning of iteration	[8](+)
	6. Producing just barely good enough artifacts (for situation at hand only)	[8](+)
Coding	1. Following coding/database/interface standards	[15, 17, 19, 24, 25, 26, 30](+)
	2. Delivering the software frequently with increments of features	[24, 25, 26, 33](+)
	3. Having customers on-site to get continuous and immediate feedback from customer for clarification	[12, 25, 27] (+)
		[24] (-)
	4. Delivering the features with high priority first	[33](+)

Copyright ⓒ 2014 SERSC

Phases	Agile based Software Development Practices	Previous studies
	5. Deploying the software gradually in real environment	[31](+)
		[15](+)
	6. Giving authority to team members to make changes at any part of the code	[17, 19, 24](+)
	7. Implementing pair programming (two programmers working together)	[19, 24, 27, 30, 59, 68](+)
		[15, 17](+)
	8. Implementing test driven development (TDD): write tests first, then write the code to pass the tests	[19, 34, 35, 72, 78] (+)
		[16, 17, 30] (-)
	9. Integrating the newly produced code to system baseline frequently	[15, 16, 17, 19, 24, 30](+)
	10. Determining code integration strategy and revising it	[36](+)
	11. Producing deliverable documentation late	[8](+)
	12. Refactoring the code and database	[28, 29, 34] (+)
		[37] (-)
Testing	1. Using acceptance tests to validate and verify user's requirements	[22](+)
	2. Implementing user interface testing	[15, 44] (+)
	3. Implementing database regression testing	[34](+)
	4. Producing executable specification	[8](+)
	5. Implementing automated tests	[15, 19, 38, 44](+)

 Copyright ⓒ 2014 SERSC

Phases	Agile based Software Development Practices	Previous studies
	6. Implementing tests continuously throughout the development	[15, 44] (+)
	7. Implementing frequent integration testing	[24, 25, 33](+)
	8. Acceptance tests are written or at least modeled by customers	[22, 39] (+)
Project Management	1. Performing project planning jointly and continuously with team members	[22, 44] (+)
		[17] (-)
	2. Conducting continuous review meetings at end of each iteration to demonstrate the latest version of software	[22](+)
	3. Carrying out release meeting at the beginning of project to plan releases	[26] (+)
	4. Carrying out iteration meeting at the beginning of each iteration to plan iterations	[26] (+)
	5. Carrying out daily stand-up meetings for daily plan	[16, 17, 26, 40] (+)
	6. Planning and estimating (cost and schedule) are based on features/ functions/stories of system	[38] (+)
	7. Conducting retrospective (postmortem) at end of each iteration to look back what worked well and what need to be improved	[20, 26] (+)
	8. Monitoring customer involvement and end-user in project activity	[36](+)
	9. Ensuring that working hours do not exceed 40 hour per week (no overtime)	[17, 24, 26](+)
	10. Revealing the current progress of iteration/sprint everyone on sprint burn down chart	[15, 16, 19](+)
Change Management	1. Controlling changes using product backlog (prioritized user stories)	[41] (+)
	2. Not allowing changes once an iteration has begin, until the iteration ends	[8, 42](+)

Copyright ⓒ 2014 SERSC

Phases	Agile based Software Development Practices	Previous studies
	3. Assigning the individual who will be responsible for ensuring Change Management activities are implemented correctly	[36](+)
	4. Automating the Change Management activities (e.g. building scripts)	[36](+)
Team	1. Emphasize on face-to-face communication	[22, 39, 43, 44, 45] (+)
	2. The team members consists of people with different functional expertise	[46] (+)
		[47] (-)
	3. Small sized team	[23] (+)
		[47] (-)
	4. Co-located team	[25, 48, 49] (+)
		[47] (-)
	5. Self-organized team	[25, 33, 50](+)
Organization	1. Encourage customer participation	[13, 25, 49, 51, 52, 53](+)
	2. Provide cooperative organizational culture instead of hierarchical	
	3. Provide facilities with proper agile-style work environment	
	4. Ensure that agile way of software development is universally accepted	

Indicators: (+): High importance, (-): Low importance

7. Limitations and Implications of the Study

This study used the purposive sampling which identified the agile software practitioners in Kedah, Penang, Kuala Lumpur and Selangor. There were a limited number of respondents (only 73 agile software practitioners participated). Therefore, a broad generalization cannot be made for Malaysia as a whole. Additionally, the results are bound to be affected by the current knowledge of the respondents, their subjective opinions and very narrow point of view from each respondent. This is because each of them has different knowledge, experience and backgrounds.

However, the results of this study are expected to give significant interest to the researcher community by providing an understanding about the implementation of agile principles and practices in the software industry. On top of that, the adoption of agile among the software practitioners is revealed.

 Copyright ⓒ 2014 SERSC

8. Conclusion

This paper has addressed the following research questions:

1. How far is the software practitioners' experience in agile based software development approach?
2. How far do the software practitioners follow the agile principles?
3. What are the agile based software development practices that are important towards producing high quality software?
4. What are the important characteristics that a team and organization should possess towards successful implementation of agile based software development?

For the first research question, it is found that the agile based software development is increasingly being implemented by the software practitioners in Malaysia (75%). However, there are still among them who have only briefly heard about it. This study concludes that the most used agile methods are XP and Scrum, while AM is the least implemented method. Furthermore, majority (96%) of the respondents agreed that agility should be considered during software development in order to produce high quality software. Among the benefits that they gained are enhancing the ability to manage changing requirements, enhancing software quality and accelerating time-to-market.

The second research question was addressed by revealing that the twelve agile principles were either 'Usually' or 'Frequently' performed by the respondents. None of them achieved 'Everytime'. This shows that agile principles were implemented by the respondents, but not constantly. This might cause the agile values are not delivered properly.

By answering the third and fourth research questions, it can be concluded that majority of the agile based software development practices and the characteristics of the agile team and organization included in the questionnaire have been agreed by the software practitioners as essential practices that need to be performed in order to produce high quality software.

Acknowledgement

The authors would like to thank the respondents who participated in this study. Additionally, special thanks go to the Ministry of Higher Education for supporting this study through the FRGS grant (S/O Code: 11889).

References

[1] A. Manifesto, "Manifesto for agile software development", [Online]. Available: www.agilemanifesto.org [Accessed: March 2, 2012].
[2] T. Dyba and T. Dingsoyr, J. Information and software technology. Empirical studies of agile software development: A systematic review,vol. 50,**(2008)**.
[3] B. W. Boehm, "Making a Difference in the Software Century",IEEE Computer,vol. 41,**(2008)**.
[4] R. Sison, S. Jarzabek, O. S. Hock, W. Rivepiboon and N. N. Hai, "Software practices in five ASEAN countries: an exploratory study", Proceedings of the 28th International Conference on Software Engineering, Shanghai, China, **(2006)** May 20-28.
[5] P. Abrahamsson, N. Oza and M. T. Siponen, "Agile Software Development Methods: A Comparative Review", Edited T. Dingsoyr, T. Dyba and N. B. Moe, Springer, New York,**(2010)**, pp. 31-59.
[6] T. Madi, Z. Dahalin and F. Baharom, "Content analysis on agile values: A perception from software practitioners", Malaysian Software Engineering Conference, Johor Baru, Malaysia, **(2011)** December 13-14.
[7] J. M. Fernandes and M. Almeida, "Classification and comparison of agile methods", Seventh International Conference on the Quality of Information and Communications Technology, Porto, Portugal, **(2010)** September 29-October 2.
[8] S. Ambler, "Agile Project Planning Tips", [Online]. Available: http://www.ambysoft.com/essays/agileProjectPlanning.html. [Accessed: January 7, 2012].

Copyright ⓒ 2014 SERSC

[9] F. Baharom, A. Deraman and A. R. Hamdan, J. ICT. A Survey on the current practices of software development process in Malaysia,vol. 4,**(2005)**.
[10] B. Solemon, S. Sahibuddin and A. A. A. Ghani, J. Advances in Software Engineering.Requirements engineering problems and practices in software companies: An industrial survey,vol. 59,**(2009)**.
[11] A. Tahir, R. Ahmad and Z. Mohd Kasirun, "An empirical study on the use of standards and procedures in software development projects", 2nd International Conference on Software Technology and Engineering, Puerto Rico, United States of America, **(2010)** October 3-5.
[12] L. Asnawi, A. M. Gravell and G. B. Wills, "Emergence of agile methods: perceptions from software practitioners in Malaysia", AGILE India, Bengaluru, India, **(2012)** February 17-19.
[13] A. L. Asnawi, A. M. Gravell and G. B. Wills, "Factor analysis: Investigating important aspects for agile adoption in Malaysia", AGILE India, Bengaluru, India, **(2012)** February 17-19.
[14] M. Omar, S.-L. Syed-Abdullah and A. Yasin, J. American Journal of Economics and Business Administration. The impact of agile approach on software engineering teams,vol. 3, no. 1,**(2011)**.
[15] VersionOne, State of agile survey. [Online]. Available: http://www.versionone.com/pdf/2011_State_of_Agile_Development_Survey_Results.pdf. [Accessed: May 10, 2012].
[16] D. West and T. Grant, "Agile development: Mainstream adoption has changed agility", Forrester Research, **(2010)**.
[17] O. Salo and P. Abrahamsson, "Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum",Software,vol. 2,**(2008)**.
[18] M. de Azevedo Santos, P. H. de Souza Bermejo, M. S. de Oliveira and A. O. m. Tonelli, J. Software Engineering & Applications. Agile Practices: An Assessment of Perception of Value of Professionals on the Quality Criteria in Performance of Projects,vol. 4,**(2011)**.
[19] L. Williams, K. Rubin and M. Cohn, "Driving Process Improvement via Comparative Agility Assessment", Agile Conference, Orlando, Florida, **(2010)** August9-13.
[20] N. Abbas, A. M. Gravell and G. B. Wills, "The Impact of Organization, Project and Governance Variables on Software Quality and Project Success", Agile Conference, Orlando, Florida, **(2010)** August 9-13.
[21] B. Ramesh, L. Cao and R. Baskerville, J. Information Systems. Agile requirements engineering practices and challenges: an empirical study,vol. 20, no. 5,**(2010)**.
[22] L. Cao and B. Ramesh, "Agile requirements engineering practices: An empirical study",Software IEEE,vol. 25, **(2008)**.
[23] D. O 'Sheedy and S. Sankaran, J. International Technology Management Review. Agile Project Management for IT Projects in SMEs: A Framework and Success Factors, vol.3, **(2013)**.
[24] B. Rumpe and A. Schroder, "Quantitative survey on extreme programming projects",Third International Conference on Extreme Programming and Flexible Processes in Software Engineering, Alghero, Italy,**(2002)** May 26-30.
[25] T. Chow and D.-B. Cao, J. Systems and Software. A survey study of critical success factors in agile software projects,vol. 81,no. 6,**(2008)**.
[26] R. Sison and T. Yang, "Use of Agile Methods and Practices in the Philippines",14th Asia-Pacific Software Engineering Conference, Nagoya, Japan, **(2007)** December 4-7.
[27] B. Tessem, "Experiences in learning xp practices: A qualitative study", M. Marchesi and G. Succi, Springer, New York,vol. 2675,**(2003)**, pp. 131-137.
[28] R. Moser, P. Abrahamsson, W. Pedrycz, A. Sillitti and G. Succi, "A case study on the impact of refactoring on quality and productivity in an agile team", Edited B. Meyer, J. R. Nawrocki and B. Walter, Springer, New York,vol. 5082,**(2008)**, pp. 252-266.
[29] M. Fowler, "Refactoring: improving the design of existing code", Addison-Wesley Profesional, Canada,**(1999)**.
[30] A. Begel and N. Nagappan, "Pair programming: what's in it for me?",Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Kaiserslautern, Germany, **(2008)** October 9-8.
[31] L. Williams and H. Erdogmus, "On the economic feasibility of pair programming", International Workshop on Economics-Driven Software Engineering Research EDSER, **(2002)**.
[32] G. Canfora, A. Cimitile, F. Garcia, M. Piattini and C. A. Visaggio, "Evaluating advantages of test driven development: a controlled experiment with professionals",Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, Rio de Janeiro, Brazil, **(2006)** September 21-22.
[33] A. C. s. C. Franca, F. Q. B. da Silva and L. M. R. de Sousa Mariz, "An empirical study on the relationship between the use of agile practices and the success of Scrum projects",Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Bolzano, Italy, **(2010)** September 16-17.
[34] S. W. Ambler andJ. Dr. Dobb's,"Survey says: Agile works in practice",vol. 31, no. 9,**(2006)**.

 Copyright ⓒ 2014 SERSC

[35] N. Nagappan, E. M. Maximilien, T. Bhat and L. Williams, J. Empirical Software Engineering. Realizing quality improvement through test driven development: results and experiences of four industrial teams,vol. 13,**(2008)**.
[36] CMMI Product Team, CMMI for Development V1.3 Technical Report, **(2010)**.
[37] M. Alshayeb, J. Information and software technology. Empirical investigation of refactoring effect on software quality,vol. 51, no. 9,**(2009)**.
[38] D. Wells,"Extreme programming: A gentle introduction", [Online]. Available: http://www.extremeprogramming.org [Accessed: September 17, 2012].
[39] F. Paetsch, A. Eberlein and F. Maurer, "Requirements engineering and agile software development", IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, **(2003)**.
[40] J. Li, N. B. Moe and T. Dyba, "Transition from a plan-driven process to Scrum: a longitudinal case study on software quality",Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Bolzano, Italy, **(2010)** September 16-17.
[41] J. Koskela, "Software configuration management in agile methods", VTT Technical Research Center of Finland,**(2003)**.
[42] J. Blankenship, M. Bussa and S. Millett, "Managing agile projects with scrum", Edited Millett, Scott, Blankenship, Jerrel, Bussa, Matthew, Springer, New York,**(2011)**, pp. 13-27.
[43] M. Coram and S. Bohner, "The impact of agile methods on software project management", 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, Greenbelt, Maryland, **(2005)** April 4-7.
[44] L. Jun, W. Qiuzhen and G. Lin, "Application of agile requirement engineering in modest-sized information systems development", Second World Congress on Software Engineering, Wuhan, China,**(2010)** December 19-20.
[45] K. N. Rao, G. K. Naidu and P. Chakka, J. International Journal of Software Engineering & Its Applications. A Study of the Agile Software Development Methods, Applicability and Implications in Industry,vol. 5,no. 2,**(2011)**.
[46] M. Cohn and D. Ford, "Introducing an agile process to an organization",Computer, vol. 36, **(2003)**.
[47] S. C. Misra, V. Kumar and U. Kumar, J. Systems and Software. Identifying some important success factors in adopting agile software development practices,vol. 82, no. 11,**(2009)**.
[48] D. Parsons, H. Ryu and R. Lal, "The impact of methods and techniques on outcomes from agile software development projects", T. McMaster, D. Wastell, E. Ferneley and J. I. DeGross, Springer, New York,vol. 235,**(2007)**, pp. 235-249.
[49] M. Lindvall, V. Basili, B. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. Williams and M. Zelkowitz, "Empirical findings in agile methods, Extreme Programming and Agile Methods", XP/Agile Universe: Springer, **(2002)**.
[50] N. B. Moe, T. Dingsoyr and T. Dyba, "Understanding self-organizing teams in agile software development", 19th Australian Conference on Software Engineering, Perth, Australia, **(2008)** March 26-28.
[51] J. Sheffield and J. Lematayer, J. International Journal of Project Management.Factors associated with the software development agility of successful projects,vol.31, no. 3,**(2013)**.
[52] R. Hoda, J. Noble and S. Marshall, J. Information and Software Technology. The impact of inadequate customer collaboration on self-organizing Agile teams,vol. 53, no. 5,**(2011)**.
[53] D. E. Strode, S. L. Huff and A. Tretiakov, "The impact of organizational culture on agile method use", 42nd Hawaii International Conference on System Sciences, Waikoloa, Big Island, Hawaii, **(2009)**January 5-8.
[54] J. Erickson, K. Lyytinen and K. Siau, J. Database Management.Agile modeling, agile software development, and extreme programming: the state of research,vol. 16, no. 4,**(2005)**.
[55] H. Corbucci, A. Goldman, E. Katayama, F. Kon, C. Melo and V. Santos, "Genesis and Evolution of the Agile Movement in Brazil--Perspective from Academia and Industry", 25th Brazilian Symposium on Software Engineering, Sao Paulo, Brazil, **(2011)** September 28-30.
[56] P. Nardi, "Doing Survey Research: a Guide to Quantitative Method", Allyn & Bacon, Boston,**(2003)**.
[57] E. Teijlingen and V. Hundley,"The importance of pilot studies", [Online]. Available: http://sru.soc.surrey.ac.uk/SRU35.html. [Accessed: March 9, 2012].
[58] A. N. Oppenheim, "Questionnaire design, interviewing and attitude measurement", Continuum International Publishing Group, London,**(2000)**.
[59] C. Schindler, "Agile software development methods and practices in Austrian IT-industry: results of an empirical study", International Conference on Computational Intelligence for Modelling Control & Automation, Vienna, Austria,**(2008)** December 10-12.
[60] R. S. Pressman,Software engineering a practitioner's approach 7th Ed., McGraw HillEducation, New York,**(2010)**.
[61] H. Glazer, "Love and marriage: CMMI and agile need each other",CrossTalk, vol. 23,**(2010)**.

Copyright ⓒ 2014 SERSC

[62] W. Zikmund, B. Babin, J. Carr and M. Griffin, "Business research methods", South Western Cengage Learning, Sydney,**(2010)**.
[63] P. Schuh, "Integrating agile development in the real world", Hingham: Charles River Media,**(2005)**.
[64] J. F. Abrantes and G. H. Travassos, "Common agile practices in software processes", International Symposium on Empirical Software Engineering and Measurement, Banff, AB, **(2011)** September 22-23.
[65] S. Nerur, R. Mahapatra and G. Mangalaraj, "Challenges of migrating to agile methodologies",Communications of the ACM,vol. 48, **(2005)**.
[66] Agile Alliance, Continuous deployment, [Online]. Available: http://guide.agilealliance.org/guide/cd.html [Accessed: January 10, 2014].
[67] P. Sfetsos, I. Stamelos, L. Angelis and I. Deligiannis, J. Empirical Software Engineering. An experimental investigation of personality types impact on pair effectiveness in pair programming,vol. 14,**(2009)**.
[68] G. Canfora, A. Cimitile, F. Garcia, M. Piattini and C. A. Visaggio, J. Systems and Software. Evaluating performances of pair designing in industry,vol. 80,**(2007)**.
[69] L. Huang and M. Holcombe, J. Information and Software Technology. Empirical investigation towards the effectiveness of Test First programming,vol. 51,**(2009)**.
[70] A. Gupta and P. Jalote, "An experimental evaluation of the effectiveness and efficiency of the test driven development", First International Symposium on Empirical Software Engineering and Measurement, Madrid, Spain, **(2007)** September 20-21.
[71] C. Desai, D. S. Janzen and J. Clements, "Implications of integrating test-driven development into CS1/CS2 curricula", ACM SIGCSE Bulletin,**(2009)**.
[72] J. C. Sanchez, L. Williams and E. M. Maximilien, "On the sustained use of a test-driven development practice at IBM", Agile Conference, Washington DC, United States of America, **(2007)** August 13-17.
[73] A. Cockburn and J. Highsmith,"Agile software development, the people factor",Computer,vol. 34, **(2001)**.
[74] M. Sliger and S. Broderick, "The software project manager's bridge to agility", Addison-Wesley, Upper Saddle River,**(2008)**.
[75] V. E. Jyothi and K. N. Rao, J. International Journal of Advanced Computer Science and Applications. Effective implementation of agile practices,vol. 2, no. 3,**(2011)**.
[76] R. A. Guzzo and M. W. Dickson, J. Annual review of psychology. Teams in organizations: Recent research on performance and effectiveness, vol. 47,**(1996)**.
[77] J. Kirakowski, "Questionnaires in usability engineering", [Online]. Available: http://www.ucc.ie/hfrg/resources/qfaq1.html [Accessed: March 8, 2012].
[78] B. George and L. Williams, J. Information and software Technology. A structured experiment of test-driven development,vol. 46,**(2004)**.
[79] L. Williams,"What agile teams think of agile principles",Communications of the ACM, vol. 55, no. 4,**(2012)**.

Authors

Aziz Deraman is a senior professor in Universiti Malaysia Terengganu (UMT). He received his Bachelor's Degree from UKM (Malaysia) in 1982, Master from Glasgow University in 1984, and PhD from UMIST in 1992. He began his career at Universiti Kebangsaan Malaysia in 1984 and is presently a senior professor of Software Engineering specializing in software process, software management and software quality. He has held various academic and administrative positions such as head of Computer Science Department (1985-1988), Deputy Dean of IT Faculty (1992-1995), Deputy Director of Computer Centre, UKM (1995-2001), Dean of the Faculty of Information Science and Technology (2001-2007), Deputy Vice Chancellor (Academic & International) of UMT and also the Vice Chancellor of the same university (2007-2012). Currently he is the dean for the School of Informatics and Applied Mathematics (UMT). His email address is a.d@umt.edu.my.

 Copyright ⓒ 2014 SERSC

Fauziah Baharom is an associate professor in UUM College of Arts and Sciences (School of Computing), Northern University of Malaysia (UUM) Kedah, Malaysia. She received her Bachelor of Science from University of Technology Malaysia (1992), Master of Science in Software Systems Technology from University of Sheffileds, UK (1995) and PhD in Computer Science from National University of Malaysia - UKM in 2008. Her research interests are software process certification, software quality and software process evaluation. Her email address is fauziah@uum.edu.my.

Shafinah Farvin Packeer Mohamed is a postgraduate student in UUM College of Arts and Sciences (School of Computing), Northern University of Malaysia (UUM), Kedah, Malaysia. She received her Bachelor of Information Technology from the same university in 2007 and now pursuing her studies in Phd. Her research interests are software certification, software quality, software process, software process evaluation, agile and secured based software development and Multi Criteria Decision Making. Her email address is shafinah@uum.edu.my.

Copyright ⓒ 2014 SERSC

Copyright ⓒ 2014 SERSC