The Compatibility and Conflict between XP Method and Level Two of CMMI-Dev1.2

Moath Husni
Information Technology
The World Islamic Sciences and Education University

Omar Tarawneh
Information Technology Department, Al-Zahra College for Women

Mejhem Al-tarawneh

Ali Naimat
Information Technology
The World Islamic Sciences and Education University

ABSTRACT
Capability Maturity Model Integration for Development Version 1.2 (CMMI-Dev1.2) is a common and popular model for controlling project development process, improving quality, and capacity evaluation. On the other side, eXtreme Programming (XP) is the one of the most popular and effective agile development method to be used for Small Software Development Firms (SSDFs). Furthermore, XP is a lightweight method that helps SSDFs in implementing the Software Process Improvement (SPI) activities as it is the more compatible method for SPI models and standards such as CMMI-Dev1.2. This paper discusses the compatibility of XP practices to the level two of CMMI-Dev1.2.

Keywords
CMMI-Dev1.2, XP Method

1. INTRODUCTION
The quality of software development processes has an essential effect on the quality of the software product. As such, software industry has realized that SPI models and standards such as CMM, CMMI, SPICE, BOOTSRAP, ISO-9000 series, and SPICE are very important in order to achieve high quality software products [1] [2][29].

CMMI has become increasingly imperative to all aspects of the software industry [3][4]. CMMI-Dev1.2 was developed specifically to guide the software development companies' for improving their processes [5], and is the most compliant with relevant SPI models and standards [6]. In addition, this model is fruitful for describing the weaknesses of the development processes in SSDFs that need instantaneously attention and improvement, particularly with agile development methods [3][6].

Agile methods are a lightweight development method that concentrate on small team size companies[7][30][31]. XP is the most popular and effective method in the development side compared to other agile methods such as SCRUM [8][9][30]. In this respect, Dyba and Dingsoyr [10] reported that 79% of the empirical reports focused on the use of the XP or SCRUM methods in general, where 76% of the reports related to use of the XP and only 3% to SCRUM practices. In addition, XP method can help SSDFs in the implementation of SPI, and they believed that XP achieves SPI better than other agile methods as it cover most of the Key Process Areas (KPAs) in level two of CMMI, while SCRUM only conforms to level one in CMMI [3] [11][31].

The overlaps and conflicts between XP method and the KPAs of CMM/ CMMI (CMMs) had been discussed by several researchers; however there are divergences in their results. These divergences came from the different manners used in alignments, where Koch [12], Paulk [13], and Omran [14] used the main objective of each KPA as a main item to do the alignment, while Martinsson [15], Elshafey and Galal-Edeen [16], and Fritzsche and Keil [17] used the specific goals of each KPA as main items to do the alignment. Therefore, there is a lack of the comprehensive and systematic alignment of XP practices to CMMs models [18].

This paper presents the comprehensive alignment of the XP practices to the specific goals of the level two of CMMI-Dev1.2 KPAs, taking into account the achievement of the specific practices of each specific goal by the same or a different way of CMMI-Dev1.2 to determine the coverage goals of each KPA by XP

2. CMMI-Dev1.2
CMMs models cover the essential elements of effective processes that use one or many disciplines to define the improvement way from ad-hoc manners toward mature and improved quality processes [19]. As shown in Figure1, CMM for Software V1.1 (1993) is the first release of CMMs, while CMMI-Dev1.3 is the newest release of CMMs, which was developed to ensure consistency among all three models and improve high maturity material in the previous generation of CMMs models. In addition, Figure 1 shows that the CMMI is a collection of previous CMMs models to demonstrate the problem of using multiple CMMs. This had been done by integrating CMMs into a single improvement framework to be used by organizations in their pursuit of enterprise-wide process improvement.

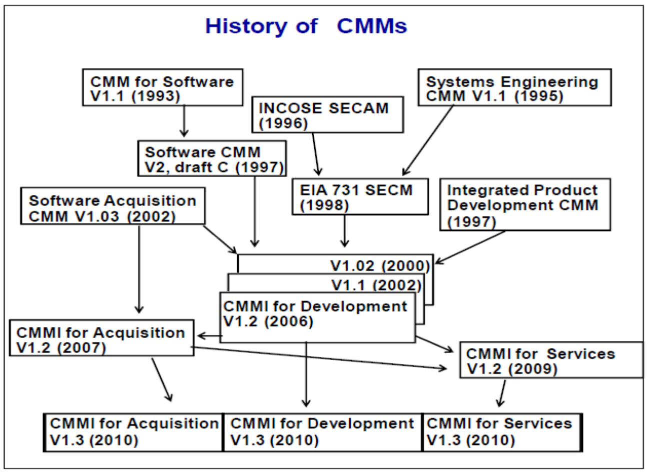
History of CMMs
CMM for Software V1.1 (1993)
INCOSE SECAM (1996)
Systems Engineering CMM V1.1 (1995)
Software CMM V2, draft C (1997)
Software Acquisition CMM V1.03 (2002)
EIA 731 SECM (1998)
Integrated Product Development CMM (1997)
V1.02 (2000)
V1.1 (2002)
CMMI for Development V1.2 (2006)
CMMI for Acquisition V1.2 (2007)
CMMI for Services V1.2 (2009)
CMMI for Acquisition V1.3 (2010)
CMMI for Development V1.3 (2010)
CMMI for Services V1.3 (2010)

Fig 1. History of CMMs (adopted from [19]).

Nowadays, CMMI becomes essential to software industry and it is necessary to inspire practitioners to adopt this model[3][4]. Accordingly, this model is common and widely used for improving process capability all over the world. Based on that, the confidence in CMMI increased as it has broad explanations of how the several best practices suitable to be used together [20][21]. In addition, this model conforms with the other SPI models such as CMM, SPICE, and ISO 9000 [22] [23].

As shown in Figure 1, CMMI-Dev1.3 is the latest version of the CMMI generations, however the KPAs of CMMI-Dev1.2 have been chosen in this paper because this version is common and widely used for assessing and improving the organizational maturity and process capability of most software development firms in the world [3][24]. In addition, CMMI-Dev1.3 is a new release and the usage of this version is still uncommon and the KPAs of CMMI-Dev1.2 are similar to level two of CMMI-Dev1.3 [19][22][25]. Furthermore, this paper focuses on level two of CMMI-Dev1.2, where the KPAs of this level are similar to level two of CMMI-Dev1.3 [19] [22].

CMMI-Dev1.2 level two considered the process as managed if it fulfills the basic infrastructure that support the development process. In addition, the managed process is a well-planned process that executed in harmony with policy; takes into account the resources, time and budget constraints to gain the expected outputs[22].

3. XP METHOD

XP method is developed by Beck [7] and it is considered one of the most popular agile method [26][30][31]. In addition, this method is commonly used in the software industry to reduce time and deal with high changing requirements environment[27].

Even though both of XP and SCRUM methods are the two popular and effective agile development methods [7] [28][30], just XP is selected in this study, as it is considered more compatible software development method for CMMI model compared to SCRUM [17].

XP method has twelve practices to increase productivity and maintaining quality. These practices are [7] [26]: Planning game, Small releases, Metaphor, Simple design, Test-Driven Development, Re-factoring, Pair programming, Collective ownership, Continuous integration, Sustainable pace (40-hour weeks), On-Site customer and Coding standards.

4. ALIGNING XP PRACTICES TO THE KPAS OF CMMI-DEV1.2 (LEVEL TWO)

The specific goals of each CMMI-Dev1.2 KPAs were used in this alignment as the main items to define the coverage ratio of these goals by the XP practices. In this regard, the descriptions of CMMI-Dev1.2 [22] and XP method [7] [26] were used as main references to explain the compatibility and conflict between XP practices and the KPAs of the CMMI-Dev1.2 (level two).

In order to perform this alignment, it is pivotal to define the suitable scales of supporting the XP practices to the KPAs. Accordingly, three scales were chosen to do the alignment, as several related studies used these scales such as [12] [13] [14]. These scales are:

- Largely Supported (L.S): the specific goals of the KPA that largely supported by XP practices.
- Partially Supported (P.S): the specific goals of the KPA that partially or implicitly supported by XP practices.
- Not-Supported (N.S): the specific goals of the KPA that are not applicable by XP practices.

Sections 4.1 to 4.7 show the alignment of the XP practices with the KPAs of CMMI-Dev1.2 (level two).

4.1 Requirements Management (P.S)

This process area aims to manage the requirements and identify conflicts between requirements and the project plan. This KPA has one specific goal:

- **S.G 1: Manage Requirements**

This specific goal consists of five specific practices, which are: (1) understand the requirements; (2) obtain commitment to requirements; (3) manage requirements changes; (4) maintain bidirectional traceability of requirements; and (5) identify inconsistencies between project work and requirements.

In XP it is well-know that requirements are collected using user story cards that are written by customer, where each card contains one feature. The programmers divide these features into two tasks: (1) customer tasks: include the scope of the project, features priority, composition of releases, releases dates; and (2) programmer tasks: include features estimations, technical consequences, process, and comprehensive scheduling. Moreover, the understanding of these requirements can be achieved by keeping the customer on-site.

The customer involvement with the development team helps in identifying the contents of each release. Therefore, iteration to release enables the customer to identify and change requirements, because small releases help to take the feedback from the customer expectations and needs. In addition, metaphor and user stories support the collaboration between the customer and developers to check the status of the requirements.

Small releases help to conduct the consistency between the requirements and other work products. In addition, user stories, functional test, and unit test help in detecting the conflicts and the inconsistencies between the project work and the requirements. However, traceability of the requirements is not largely supported by XP, because there is no data repository in XP to save and trace story cards and make the documentation up-to-date.

As concluded, the specific goal the requirement management KPA is partially supported by some of XP practices such as on-site customer, planning game, continuous integration, metaphor, and small releases. However, XP does not support a repository that save and trace user stories.

4.2 Project Planning (L.S)

This process area aims to create and maintain plans that define all the project activities. This KPA consists of three specific goals:

- **S. G 1: Establish Estimates**

This specific goal consists of four specific practices, which are: (1) estimate the project scope; (2) estimate the product and task attributes; (3) define project life cycle; and (4) estimate the effort and cost.

In XP, the planning game practice makes the customer able to select the features to be developed for the next release; then the programmers will divide the features into tasks and estimate them. The customer role is to define the project scope and prioritize the features where the programmer's tasks concentrate more on: estimations of the features, process, and time scheduling.

In addition, XP the development team involved in early planning by estimating the effort needed to implement user stories. Therefore, the estimated tasks are established and may be updated during the process. Furthermore, the iteration to release practice helps to increase the estimation precision.

- **S. G 2: Develop a Project Plan**

This specific goal consists of seven specific practices, which are: (1) plan for budget and schedule; (2) identify project risks; (3) plan for data management; (4) plan for project resources; (5) plan for needed knowledge and skills; (6) identify stakeholder involvement; and (7) establish the project plan.

In XP exploration phase, the developers explore architectural prototype and test the technology to be used in developing the prototype. In addition, collective ownership practice guarantees the involvement of all stakeholders that help in increasing the commitment and obligation to the iteration plans.

Based on the project plan, risks are defined, the team training program is planned, and the involvement of all the team is guaranteed. In addition, incremental XP life cycle helps the developers to identify and manage risks efficiently. Furthermore, the planning game practice is used for establishing the project schedule, budget, and plan for each iteration.

- **S. G 3: Obtain Commitment to the Plan**

This specific goal consists of three specific practices, which are: (1) review project plans; (2) reconcile progress and resource levels; and (3) assure plan commitment.

In XP, the commitment to the release and plans can be attained by assuring the involvement of any team member based on his\her role and responsibility. In addition, the tracker traces and monitor the progress of each iteration and evaluates whether it achieves the goal within the budget and time constraints. Furthermore, the coach is responsible to ensure that the project performed correctly by keeping the development team implementing the selected features for the actual iteration.

As concluded, the specific goals of the project planning KPA are largely supported by some of XP practices such as planning game, small releases, on-site customer, and metaphor.

4.3 Project Monitoring and Control (L.S)

This process area aims to monitor the project's progress and take the suitable actions that keep the project's performance on the right path. This KPA consists of two specific goals:

- **S. G 1: Monitor Project Against Plan**

This specific goal consists of seven specific practices, which are: (1) monitor project planning parameters; (2) monitor commitments; (3) monitor risks; (4) monitor data management; (5) monitor stakeholder involvement; (6) review progress; and (7) review milestone.

In XP method, The tracker monitors the schedule, traces the estimates that have been created by the development team and provides feedback to improve the future estimations. Furthermore, the tracker is responsible for calculating project performance metrics during the iteration. Spreadsheet tool is a commonly used in XP projects for calculating metrics such as estimates and actual achievements.

Using the big visual chart and calculating the project velocity (the number of stories of a given size that developers can implement in an iteration) support the commitments of the stories during the small releases. Therefore, this commitment process clarifies roles of the customer and the project team at the tactical level, and makes the project flexible at the strategic level. Thus, project's progress data is collected by the use of measures and the functional tests are performed to check the milestones against the schedule.

XP method enables the coordination and collaboration with relevant stakeholders by integrating developers, customer, testers, and management, using "self-organizing cross-functional team". In addition, collective ownership practice helps in integrating all the team members in the project work. Furthermore, the intensive communications between the customer and developers handle the changes that are needed during the iteration, and this can be done with assistance from the coach.

- **S. G 2: Manage Corrective Action to Closure**

This specific goal consists of three specific practices, which are: (1) analyze issues; (2) manage corrective action; and (3) take corrective action.

Short iteration and regular commitments are fruitful for monitoring and managing the project against the baseline, and also offer opportunities to make the required modifications. Therefore, the actions that should be taken in response to this modification may affect the method used, and the functionality. On the other hand, the communication between the customer and the development team helps to declare the modification and what information should be used to perform it.

Coach is responsible to ensure that the programmers are working in an efficient and effective way, and he solves programmers' problems quickly. On the other hand, tracker traces the iteration progress and evaluates that the goal is achieved, and gives feedback on how accurate the team for improving future estimations. As such, the tracker is responsible for informing the results of daily meetings to check the output of each iteration against the plan.

Furthermore, the big visual chart also supports this specific goal, where the project and stories of the small releases are clearly stated. This visual chart is commonly created by the customer and development team.

As concluded, the specific goals of the project monitoring and control KPA are largely supported by some of XP practices such as on-site customer, test-driven development, collective ownership, and small releases.

4.4 Supplier Agreement Management (N.S)

This process area aims to manage the achievement of products from suppliers using a formal agreement.

Several studies [14] [15] [17] stated that this KPA is not supported by XP. In this respect, Fritzsche and Keil [17] claimed that this KPA is not addressed by XP, as the involving suppliers could be problematic; but they believe that the XP can be improved to satisfy the goals of this process area with keeping the agility of XP method. In addition, Omran [14] stated that this KPA seems to consume significant resources from small teams.

Therefore, this KPA is not supported by XP and there is a need to extend XP to meet this process area and keep the XP agility values.

4.5 Measurement and Analysis (P.S)

This process area aims to establish a measurement capability that satisfies the management information needs. This KPA consists of two specific goals:

- **S. G 1: Align Measurement and Analysis Activities**

This specific goal consists of four specific practices, which are: (1) establish measurement objectives; (2) specify measures; (3) specify data collection and storage procedures; and (4) specify analysis procedures.

In XP, the project metric is recommended. Furthermore, tracker defines the measurements and analysis procedures the based on: (1) tracing the estimates that have been created by the development team and he provides feedback to improve future estimations, and the tracker is not advised to interrupt the project frequently; and (2) tracing the progress of each iteration and evaluates whether the goal can be achieved within the time and resources constraints, or define the changes that may require in the process.

- **S. G 2: Provide Measurement Results**

This specific goal consists of four specific practices, which are: (1) collect, analyze, store measurement data and (2) communicate results.

In XP, the tracker collects the project's progress data by estimating the project velocity, and he develops the programmers feedback by asking and listening to what they are doing currently. Consequently, the intensive communications between the development team and the customer can help to transfer the important data to measurement results to be used from the team members. In addition, functional test is used to check the milestones against the schedule. Furthermore, the tracker uses wall charts to convey the results of analyzing the measurement data.

As concluded, the measurement and analysis KPA specific goals are partially supported by on-site customer and test driven development practices. However, XP does not support a data repository to save and retrieve the measurement data.

4.6 Process and Product Quality Assurance (P.S)

This process area aims to provide an objective insight into processes and associated work products for the staff and management. This KPA consists of two specific goals:

- **S. G 1. Objectively Evaluate Processes and Work Products**

This specific goal consists of two specific practices, which are: (1) objectively evaluate processes; and (2) objectively evaluate work products and services.

The planning for quality assurance's activities is clearly satisfied by pair programming, continuous integration, and test driven development practices. In addition, the regular programming sessions focus on the quality. Furthermore, Coach is responsible for guiding the team to perform XP method in the right way. Accordingly, the quality issues can be easily resolved by XP team and customer.

- **S. G 2: Provide Objective Insight**

This specific goal consists of two specific practices, which are: (1) communicate and ensure resolution of noncompliance issues; and (2) establish records.

The customer can assure the correctness of the systems when all functional tests are performed successfully. Consequently, the application to be developed is evolving iteratively in parallel with performing the quality assurance activities. In addition, quality assurance results are commonly presented in a graphical way to be used by the project team such as the results of test-failures of each release.

As concluded, the specific goals of the process and product quality assurance KPA are partially supported by some of XP practices such as continuous integration, test driven development, and pair programming practices. However, XP method does not support an evaluation of the quality of processes, products and services against the applicable process descriptions. In addition, there are no strict and clear guidelines for resolving issues and for creating records that related to the activities of quality assurance.

4.7 Configuration Management (L.S)

This process area aims to establish and maintain the software product integrity using configuration identification, control, status accounting, and audits. This KPA consists of three specific goals:

- **S. G 1: Establish Baselines**

This specific goal consists of three specific practices, which are: (1) identify configuration items; (2) establish a configuration management system; and (3) create or release baselines.

Code, design, tests and requirements are considered the steps of configuration in XP. In addition, the iteration to releases give a strong baselines mechanism and careful version control of the code and other release components. Furthermore, the using of a configuration management system is covered by continuous integration, collective ownership, and small releases. Moreover, the release baselines are always established using the functional tests and at the end of each iteration.

- **S. G 2. Track and Control Changes**

This specific goal consists of two specific practices, which are: (1) track change requests; and (2) control configuration items.

Pair programming, tests, and on-site customer feedback are used for tracking and controlling the changes. Moreover, re-factoring practice pushes the program source code in the direction of a larger baseline, with more classes and methods in common.

- **S. G 3: Establish Integrity**

This specific goal consists of two specific practices, which are: (1) establish configuration management records; and (2) perform configuration audits.

The continuous integration practice increases the project velocity to reach the production state, where the changes that have been made by a pair of programmers should not affect other component that were developed by another pair of programmers. In addition, coding standard practice keeps the code consistent and easy to read, which makes the development team able to understand all the code chunks to be developed as the basis for the collective ownership practice. Therefore, the code developed based on the coding standard practice that means it has specific descriptions. On the other hand, pair programming, pair programming, on-site customer and test driven development practices are informally used for performing the auditng.

As concluded, the specific goals of the configuration management KPA are largely supported by some of XP practices such as planning game, continuous integration, re-factoring, on-site customer, test-driven development, coding standard, collective ownership, and small releases.

As a result of this alignment, the following can be concluded:

- Three KPAs that largely supported by XP are: (1) project planning; (2) project monitoring and control; and (3) configuration management.
- Three KPAs that partially supported by XP are: (1) requirement management; (2) measurement and analysis; and (3) process and product quality assurance.
- One KPA is not-supported by XP namely, supplier agreement management.

5. THE SIMILARITIES AND DIFFERENCES OF THIS ALIGNMENT WITH THE RELATED STUDIES

Several studies [14] [16] [17] discussed to identify what the CMMI-Dev1.2 KPAs that can be covered by XP. In conducting the coverage of XP practices to the CMMI-Dev1.2 KPAs by these studies, five scales were used by Fritzsche and Keil [17], while three scales were used by Omran [14], and Elshafey and Galal-Edeen [16].

As shown in Table 1, the descriptions of the used scales by the related studies focus on common three scales, which are: (1) Largely Support (L.S): XP practices largely support the specific goals of the KPA; (2) Partially Support (P.S): XP practices partially support the specific goals of the KPA; and (3) Not-Support (N.S): XP practices do not support or are not applicable for the specific goals of the KPA. Based on these common three levels, Table 2 unites the different scales used by these studies into common three scales used in doing the alignment of this paper. Accordingly, Table 3 shows the alignment results of the three studies based on the common three scales.

Table 1. Scales of other studies in coverage XP practices to CMMI-Dev1.2 KPAs

Related Studies	Scale of Comparison
[17]	• Conflicting (–): XP practices cannot cover the process area's components • Not addressed (0): XP practices do not cover the process area's components. • Partially supported (+): XP practices satisfy some of the process area's components. • Supported (++): XP practices satisfy most of the process area's components. • Largely supported (+++): XP practices satisfy the major part of the process area's components.
[14]	• (++): process area is largely addressed by XP practices. • (+): process area is partially addressed by XP practices. • (--): process area is not addressed by XP practices.
[16]	• Supported (S): when most parts of the process area is supported by XP practices that will help enhance or accelerate its implementation. • Partially Supported (P.S): when only a small part of the process area is covered by an XP practice, it can't help implementing this process area on its own other non XP practices will be needed. • Not-Supported (N.S): when process area is not addressed by XP method.

Table 2. Unification the used scales of other studies in three scales

Common Scales	Related Studies		
	[17]	**[14]**	**[16]**
Largely Supported (L.S): XP practices largely support the specific goals of the KPA.	(+++)	(++)	Supported
Partially Supported (P.S): XP practices partially support the specific goals of the KPA.	(++) OR (+)	(+)	Partially Supported
Not- Supported (N.S): XP practices do not support or are not applicable for the specific goals of the KPA.	(-) OR (0)	(--)	Not-Supported

Table 3. Coverage results of XP practices to CMMI-Dev1.2 KPAs of the related studies.

CMMI-Dev1.2 KPAs (Level-Two)	Related Studies			Alignment of this Study
	[17]	[14]	[16]	
Project Planning	L.S	L.S	L.S	L.S
Project Monitoring and Control	L.S	L.S	L.S	L.S
Supplier Agreement Management	N.S	N.S	N.S	N.S
Requirement Management	L.S	L.S	L.S	P.S
Measurement and Analysis	P.S	L.S	P.S	P.S
Process and Product Quality assurance	P.S	P.S	N.S	P.S
Configuration Management	L.S	P.S	P.S	L.S

As shown in Table 3, the following can be concluded:

- Three KPAs have the same results in this study compared to other related studies. These KPAs are project planning, project monitoring and control, and supplier agreement management.
- Four KPAs have the different results in this study compared to other related studies. These KPAs are requirement management, measurement and analysis, process and product quality assurance, and configuration management.

6. CONCLUSION

CMMI-Dev1.2 was created especially for the software firms in order to improve their development and management processes. XP is the most well-known and common agile method. This paper aimed to align XP practices to the KPAs of CMMI-Dev1.2 (level two). In this alignment, three scales were used to represent the coverage ratio of supporting XP practices to the specific goals of each CMMI-Dev1.2 KPAs. These scales are largely supported, partially supported, and not-supported. The results of this alignment so that (1) three KPAs were largely supported by XP practices, namely: project planning, project monitoring and control, and configuration management; (2) three KPAs were partially supported by XP practices which are: requirement management, measurement and analysis, and process and product quality assurance; and (3) one KPA was not-supported by XP practices which is supplier agreement management KPA.

In addition, this paper discussed the similarities and differences between the alignment's results of this study compared to other related studies. This comparison shows that three KPAs of CMMI-Dev1.2 (level two) have the same results in this study compared to other related studies, while the last four KPAs have different results in this study compared to other related studies.

7. REFERENCES

[1] D. Bae, “Panel: Software Process Improvement for Small Organizations”, in the 31st Annual International Computer Software and Applications Conference, 2007, pp. 1-5.

[2] M. N. Khokhar, K. Zeshan, K., & J. Aamir, “Literature review on the software process improvement factors in the small organizations”, in the 4th International Conference on New Trends in Information Science and Service Science (NISS), 2010, pp. 592 – 598.

[3] M. Pikkarainen, “Towards a framework for improving software development process mediated with CMMI goals and agile practices, Academic Dissertation, Faculty of Science, Department of Information Processing Science, University of Oulu, Finland, 2008.

[4] F. H. Alshammari, R. Ahmad, “The effect of geographical region on the duration of CMMI-based software process improvement initiatives: An empirical study”, in the 2nd International Conference on Software Technology and Engineering (ICSTE), 2010, pp. V2-97-V2-100.

[5] T. Galinac, “Analysis of Quality Management in Modern European Software Development”, Electronic form only: NE Eng. Rev, Vol. 28, No. 2, 2008, pp. 65-76.

[6] I. Garcia, C. Pacheco, and J. Calvo-Manzano, "Using a web-based tool to define and implement software process improvement initiatives in a small industrial setting", Software, IET, vol. 4, NO 4, 2010, pp. 237-251.

[7] K. Beck, Extreme programming explained: embrace change: 3th End.Reading, Mass, addition-Wesley. Boston, 2000.

[8] J. A. H. Alegra and M. C. Bastarrica, "Implementing CMMI using a Combination of Agile Methods", CLEI ELECTRONIC JOURNAL, Vol. 9, No 1, 2006, pp. 1-15.

[9] L. Zoysa, “Software Quality Assurance in Agile and Waterfall Software Development Methodologies: A Gap Analysis”, Ph.D. thesis, School of Computing, University of COLOMBO, Sri Lanka, 2011.

[10] T. Dyba, & Dingsøyr, “Empirical studies of agile software

development: A systematic review”, Information and Software Technology, Vol. 50, No. 9-10, 2008, pp. 833-859.

[11] E. Erharuyi, “Combining eXtreme Programming with ISO 9000: 2000 to Improve Nigerian Software Development Processes”, M.S. thesis, School of Engineering, Blekinge Institute of Technology, Sweden, 2007.

[12] A. S. Koch,”CMM-compliant XP”, paper retrieved on 20 Aug. 2012, from http://www.askprocess.com/Articles/CMM-XP.pdf.

[13] M. Paulk, “Extreme Programming from a CMM Perspective”. IEEE Software, Vol. 18, No. 6, 2001, pp. 19-26.

[14] A. Omran, “AGILE CMMI from SMEs perspective” in the 3rd International Conference on Information & Communication Technologies: from Theory to Applications (ICTTA 2008), 2008, pp. 1-8.

[15] J. Martinsson, “Maturing Extreme Programming Through the CMM”, M.S. thesis, Department of Computer Science, Lund University, Lund, Sweden, 2002.

[16] L. A. Elshafey, & G. Galal-Edeen, “Combining CMMI and Agile Methods”, In the 6th International Conference

on Informatics and Systems (INFOS2008), 2008, pp. SE-27- SE-39.

[17] M. Fritzsche, and P. Keil, "Agile Methods and CMMI: Compatibility or Conflict?," e-Informatica Software Engineering Journal, Vol. 1, No. 1, 2007, pp. 9-26.

[18] M. Pikkarainen & M. Annukka, "An Approach for Using CMMI in Agile Software Development Assessments: Experiences from Three Case Studies", in the SPICE 2006 conference, 2010, pp. 1-11.

[19] C. P. Team, "CMMI for Development (CMMI-DEV): Version 1.3", Technical Report, CMU/SEI-2010-TR-033, Software Engineering Institute, Carnegie Mellon University, USA, 2010.

[20] M. Bush, & D. Dunaway, CMMI (R) Assessments: Motivating Positive Change (Sei Series in Software Engineering): Addison-Wesley Professional. Murray, KY, U.S.A. 2005.

[21] D. Goldenson, and D. Gibson, "Demonstrating the impact and benefits of CMMI: an update and preliminary results", CMU/SEI-2003-SR-009, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 2003.

[22] C. P. Team, "CMMI for Development, version 1.2", Technical Report, CMU/SEI-2006-TR-008, Software Engineering Institute, Carnegie Mellon University, USA, 2006.

[23] P. Mongkolnam, U. Silparcha, N. Waraporn, & V. Vanijja, "A Push for Software Process Improvement in Thailand", in the 16th Asia-Pacific Software Engineering Conference, 2009, pp. 475-481.

[24] D. Mishra, and A. Mishra, "Software process improvement in SMEs: A comparative view", Computer Science and Information Systems, Vol. 6, No. 1, 2009, pp. 111-140.

[25] A. B. M. Isawi, "Software Development Process Improvement for Small Palestinian Software Development", M.S. thesis, Faculty of Graduate Studies, An-Najah National University, Nablus, Palestine, 2011.

[26] R. Jeffries, A. Anderson, & C. Hendrickson, Extreme Programming Installed: Addison-Wesley. Boston, 2002.

[27] K. S. Devesh, S. C. Durg, & S. Raghuraj, "Square Model-A Proposed Software Process Model for BPO based Software Applications", International Journal of Computer Applications, Vol. 13. No.7, 2011, pp. 33-36.

[28] P. Abrahamsson, O. Salo, J. Ronkainen, & J. Warsta, "Agile software development methods", Technical Report, Espoo: VTT Publications 478, Technical Research Centre of Finland, Finland, 2002.

[29] Fuggetta A, Di Nitto E. "Software process". InProceedings of the on Future of Software Engineering 2014 May 31, pp. 1-12). ACM.

[30] M.Altarawneh, "Monitoring oriented agile based web applications development methodology for small software firms in Jordan." PhD thesis,School of computing, Universiti Utara Malaysia, 2016.

[31] E. Kouzari, V. C. Gerogiannis, I. Stamelos, and G. Kakarontzas, "Critical success factors and barriers for lightweight software process improvement in agile development: A literature review," 10th International Joint Conf. on Software Technologies (ICSOFT), vol. 1, pp. 1-9, July 2015.