Agile Maturity Model (AMM): A Software Process Improvement framework for Agile Software Development Practices

Chetankumar Patel[(1)] and Muthu Ramachandran [(2)]

(1) Innovation North, Faculty of IT, Leeds Metropolitan University (United Kingdom)
E-mail: c.patel@leedsmet.ac.uk

(2) Innovation North, Faculty of IT, Leeds Metropolitan University (United Kingdom)
E-mail: m.ramachandran@leedsmet.ac.uk

ABSTRACT

Agile software development methodologies have introduced best practices into software development. However we need to adopt and monitor those practices continuously to maximize its benefits. Our research has focused on adaptability, suitability and software maturity model called Agile Maturity Model (AMM) for agile software development environments. This paper introduces a process of adaptability assessment, suitability assessment, and improvement framework for assessing and improving agile best practices. We have also developed a web based automated tool support to assess adaptability, suitability, and improvement for agile practices.

Keywords: Software Process Improvement, Agile Maturity Model, Agile Software Development

1- INTRODUCTION

Agile software development methodology is a conceptual framework of practices and principles to develop software faster, incrementally and to produce satisfied customer. Several agile software development methodologies have been suggested in the literature, like Extreme Programming [3], Scrum [32], Crystal Methodology [12] and Mobile-D [19]. All these methods adopt agile principles, such as iterative development, frequent and early delivery of working software, and simplicity as defined in Agile Manifesto [1].

Agile methods defines how the development should be carried out under agile values and principles [1], to address the challenges like requirements change, customer satisfaction and rapid development [30]. According to [18] “Agility is the ability of to both create and respond to change in order to profit in a turbulent business environment” [18].

While on the second hand CMM(I) or software process improvement has

gained a lot of attention during the last decade. Due to the increasing competition in the software market faster delivery, high quality products and customer satisfaction are the major concerns for software organizations. A quality process can have a positive impact on services, cost, on-time delivery, development technology, quality people and quality of products [35].

It is challenging issue that the CMM, CMMI for software and process improvement were applicable to the agile methods, these two approaches have been informally characterized as having the same relationship as oil and water [34].

Agile software development methodology is the iterative software development methodology for small to medium organization and main objectives are lower cost, high productivity and satisfied customer. The CMM tends not to focus the software process on an organization's business objectives in their software process improvement programme [28]. Also most companies, small to large companies found it is too difficult to reach higher levels in the CMM [7][8] [22].

[17] reported the use of the CMM in several software organizations. The study consistently showed significant organizational performance improvements that were directly associated with process maturity. The study also mentioned that the CMM improvement path is not always smooth, the efforts generally took longer and cost more than expected. While agile software development methodology is targeted to lower cost. Some of the KPAs have been found difficult to apply in small projects [7]. This may be because CMM was originally structured for big enterprises [22]. CMM addresses practices such as document policies and procedure that large organizations need because of their size and management structure [7].

Normally agile software development practices do not support the heavy documentation at all and people communicate verbally on on-going basis. Unlike CMM, CMMI does not just focus on software process management; it also considers other department such as marketing, finance and purchasing [2]. So it could be seen unnecessarily complex, when it is applied to agile software development practices like Extreme programming, Scrum and lean development. When businesses adapt the CMMI they should be familiar with the CMM practices. CMMI Based upon the software CMM and has most of the same process areas. It may also inherit some of the same problems as CMM, such as the problem in reaching higher capability levels [5]. This is not acceptable against the agile software developments principle and motivation.

CMMI is a process oriented. Maturity of an organization is depends on the practices that are to be followed, not on the result. But Agile practices (XP) promises client satisfaction, no over time etc. thus it would not be a good idea to say that a given project is on the highest level of XP/Agile maturity while it suffers from the overtime and lacks of client satisfaction [23]. It is really important to have a process improvement framework for the agile software de-

velopment practices. CMM and CMMI and their like have been accused of being bureaucratic models that forces their client to go through their maturity ladder without ensuring that they will achieve a quality product or meet their business objectives [33]. According to [16], CMM and CMMI and their like as not a 'Good medicine' for even very large system engineering projects and they are overly complex for most IT projects.

There is a consensus that the current standard software process improvement frameworks such as CMM can not be applied unmodified to small organizations [7]. While majority of the organizations who adopt or using the agile software development practices are categorized as small to Medium enterprises. In addition, they need to be tied to the business objectives as there is no mechanism for doing this yet with the current software process improvement models [28].

Agile software development practices are more business objective oriented practices. Achieving business objectives is one of the important recipes for information technology business success [20]. Current software process improvement models have not yet shown a mechanism for aligning SPI Activities with business objectives [13] [15] [28] [33]. So it is really difficult to do mapping the current software process improvement model with agile software development practices. However Paulk [24] suggested the Extreme programming form a CMM perspective. But this approach is also difficult and inadequate to identify or define the maturity level of the organization based on his report.

There is a need for a Software process improvement model to suit agile software development environments. Therefore the purpose of this paper is to propose and evaluate a Software process improvement model for Agile Software development environments and enhance the adaptability of agile software development methodology and its practices. The purpose of this paper is to define a generic process model for software process improvement that is suitable for agile software development environments, to identify and define agile practices for each maturity level and relate agile practices problem to agile practices improvement goals

2- PROCESS IMPROVEMENT FRAMEWORK FOR AGILE SOFTWARE DEVELOPMENT PRACTICES (Agile Maturity Model (AMM))

According to Christie [11], defining processes is recognized as critical elements in the software process improvement [11]. To keep the representation clear, understandable and usable the AMM links the agile software development practices to maturity levels, but it is not an exhaustive representation of agile software development practices. The AMM model is based on the agile software development values, practices and principles.

The AMM model is designed to improve and enhance the agile software development methodology and boost up the agile principles and objectives like the lower cost, customer satisfaction, software quality, etc. Figure 1 introduces the AMM (Agile Maturity Mode)l for agile software development. This high level view of the model shows how agile software development practices mature from an initial or ad-hoc level to continuously improving level based on the agile principles and practices. In this model each level has a pre defined goal to help practitioner or organization focus on their improvement activities.

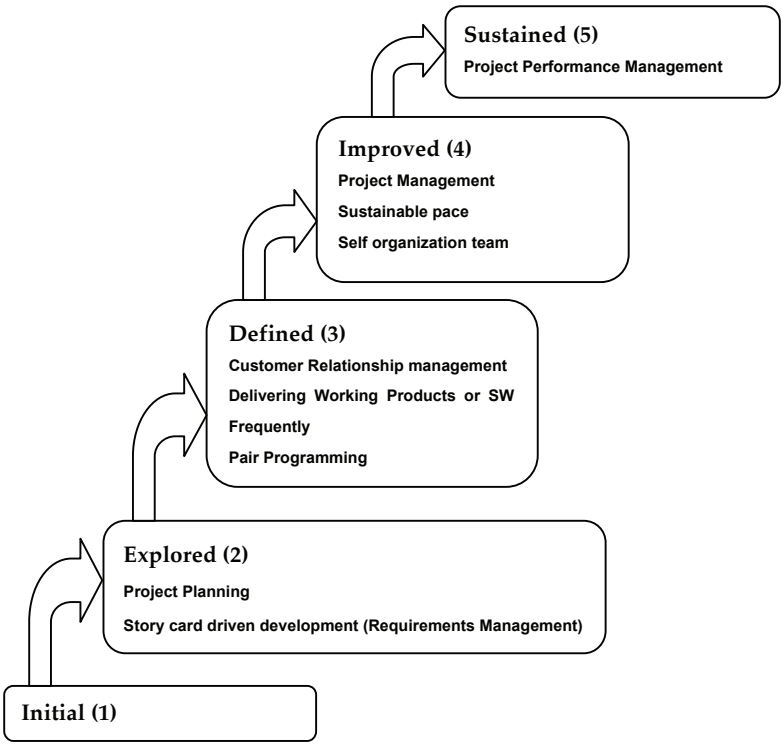

Figure 1 Agile Maturity Model (AMM)

Level 1: Initial Level (Not accommodating at all There is no process improvement goals defined at this unstructured level)

The software development practices or process is very slim at this level and not necessarily repeatable. Organizations typically do not provide a stable environment for development. Level 1 company does not have defined agile software development process. The main problems at this level relate to overtimes, schedule slips, communication, software quality and development cost. These companies operate in their own unique way and depend on particular

people rather than whole team. [25] describe for traditional software process, success at this level is depends on 'the competence and heroics of the people in the organization and cannot be repeated unless the same individuals are assigned to the next project'.

Level 2: Explored (project or software planning, customer or stakeholders orientation practices)

Level 2 denotes a more structured and complete software development practices than level 1. Organization with level 2 capability experienced fewer problems with their software development process than their level 1 counterparts.

Problems with communication and coding and integration practices remain a major problem along with staff retention. Technically, difficulty for level 2 companies centered on communication (mutual interaction), coding standards, overtimes and customer satisfaction.

The goals for this level are

- Project planning
- Improve agile requirements engineering
- Customer and stakeholders' orientation practices
- Enhance Value, Collaboration and Planning Practices

An organization at this level has focused on the project planning, the planning game practice is used to create the project plans, release planning is used to create schedules, estimation id done by the developers based on the business values supplied by the onsite customer, etc. An organization at this level has introduced policies that ensure that story cards (Requirements) are specified and used the standard structure of the story cards and story cards are written by the on-site customer. Level 2 in general denotes that an organization has devoted resources to the planning and story cards (requirements engineering) practices as a whole.

In general companies at this level 2 process capability have established the project planning, requirements engineering (Story cards driven development) and on-site customer related practices to track project schedules, plan, requirements (Story cards), cost and functionality.

The AMM at level 2 maturity aims to help developers and customers to identify and improve problems related to planning, requirements engineering and on-site customer by learning from previous project success and failures. This is achieved by an assessment of current process and to identify where weakness lie will help development team gain a general overview and allow them to address any planning or requirements issues associated with individual projects. The figure 2 summaries goals, key process areas and assessment questionnaires for AMM maturity level 2.

Level 3: Defined Level (Customer satisfaction, Software quality and development practices)

Level 3 denotes a more focus on practices related to customer relation ship management, frequent deliveries, pair programming, communication, coding, testing and quality of software. The goals for this level are as

- Customer satisfaction
- Communication improvement
- Software quality
- Enhancement on coding practices and coding standards

The customer relationship is maintained very well at this level. At this level companies ensure a deeper understanding of the test driven development for coding and testing, pair programming and subsequently enhance the ability to deliver software frequently and coding standards.
Level 3 companies had increased their control over their technical practices like coding and testing practices and furthermore the practices related to customer relation ship management is focused, but saw little improvement on managing the practices related to people. They continued to report problem on working hours or overtime management sustainable pace for development team and project management. At this level no structured risk assessment is performed. Furthermore no consideration is taken towards code optimizations. At level 3 most of the technical problems are solved but the organizational problem like problems related to the team are unsolved.

The AMM at level 3 maturity aims to help developers identify and improve problems related to customer relationship, coding, testing, frequent releases and coding standards. This is achieved by an assessment of current process and to identify where weakness lie. The table 1 summaries goals, key process areas and assessment questionnaires for AMM maturity level 3.

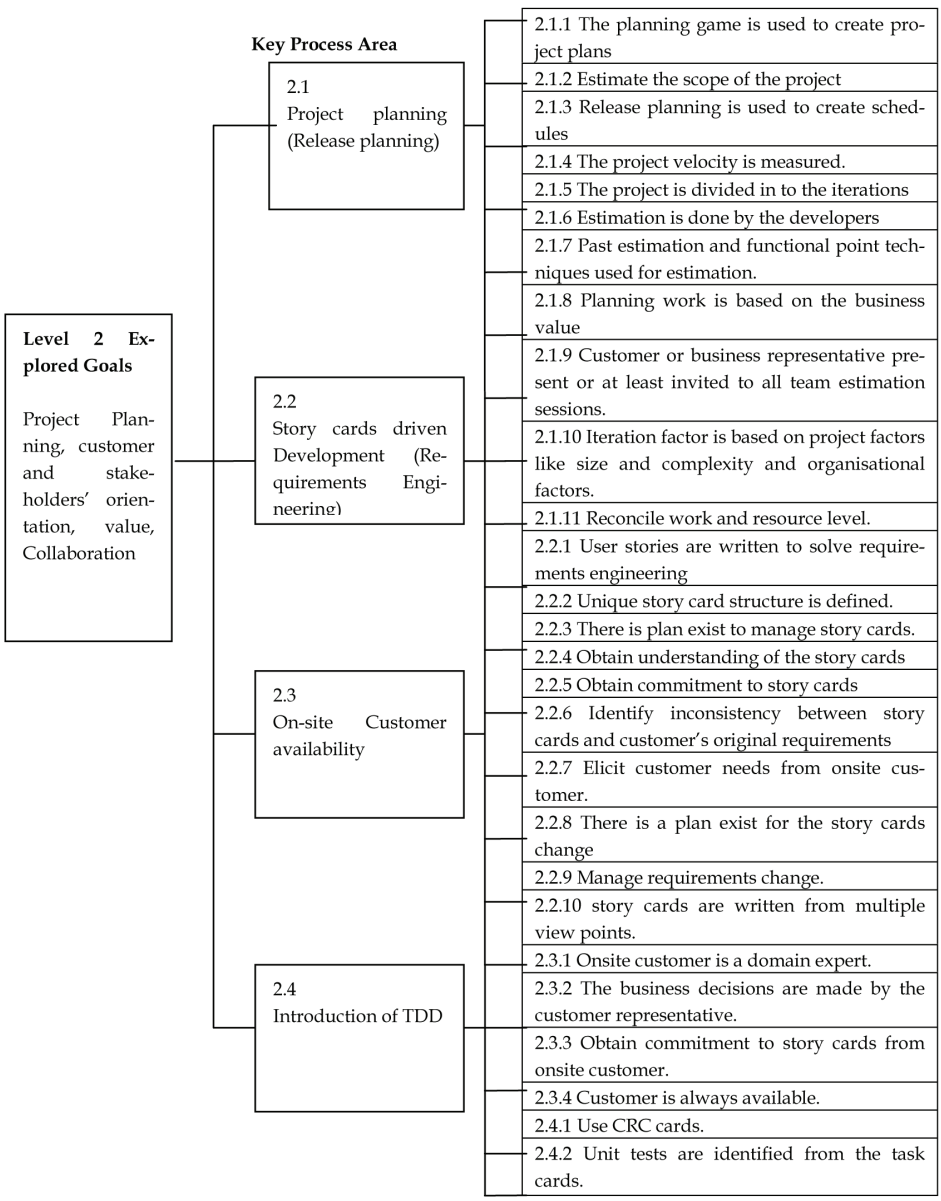

Figure 2 Goal, Key process area and assessment questionnaires for AMM Level-2

Table 1 Goal, Key process area and assessment questionnaires for AMM Level-3

Level 3: Defined Level (Customer satisfaction, Software quality and development practices)	3.1 Customer Relationship Management	3.1.1 System metaphor is defined, which allows the customer representative to understand the system (Choose system metaphor)
		3.1.2 Customer and business representative present or at least invited to all team estimation sessions
		3.1.3 Refactoring is encouraged via a little-and-often approach and larger refactoring reprioritised with the customer
		3.1.4 Make small frequent release
		3.1.5 User stories are written.
		3.1.6 Project plans are created to create project plan
		3.1.7 Effectively collaborate customer
	3.2 Delivering Working Products/SW frequently	3.2.1 Made small frequent releases which will create feedback loop
		3.2.2 Only one pair integrates code at a time
		3.2.3 Integrate often
	3.3 Pair Programming	3.3.1 move people around
		3.3.2 the customer pay frequent visit to the development team
		3.3.3 all production code is pair programmed
		3.3.4 only one pair integrates code at a time
		3.3.5 use collective code ownership
		3.3.6 overtime data are collected and published
	3.4 Mutual Interaction	3.4.1 All code is pair programmed.
		3.4.2 Story cards are written by the collaboration of on-site customer and developers
		3.4.3 Communicates result through acceptance testing
		3.4.4 Refactoring is encouraged via a little-and-often approach and larger refactoring reprioritised with the customer
		3.4.5 Team members have an open work environment that supports collaboration and conversation

	3.5 Test Driven Development	3.5.1 Code the unit test first
		3.5.2 Refactor whenever and wherever possible
		3.5.3 All code must have a unit tests
		3.5.4 All code must pass the unit test and score must be published before it can be released
		3.5.5 When a bug is found tests are created
		3.5.6 Are the best practices for automated tested encouraged, rewarded and in place on the project?
		3.5.7 Perform Peer-reviews
		3.5.8 Analyse results and identify corrective action
	3.6 Implementation and Interaction	3.6.1 No functionality is added early
		3.6.2 Integrate often
		3.6.3 Automated testing is used to support frequent integration test
		3.6.4 No up front design
		3.6.5 team delivers useful content for business review every 1-4 weeks
		3.6.6 the list of user story is reprioritised based on an updated evaluation of the project at each iteration boundary
		3.6.7 the best practices for continuous integration are encouraged, rewarded and in place on the project?
		3.6.8 Prepare for product integration
		3.6.9 Determine integration sequence
	3.7 Coding Standards	3.7.1 Codes must be written to agreed standards
		3.7.2 Code the unit test first
		3.7.3 All production code is pair programmed
		3.7.4 Only one pair integrates code at a time
		3.7.5 Use collective code ownership

Level 4: Improved (People orientation and project management Practices)

Companies at this maturity level are in a position to collect detailed measure of the software development process or practices and product quality, both the software development practices and products are quantitatively understood and controlled using detailed measurements [25].

The improved level of the AMM model is focused on the project management, working hours, self organising team, risk assessment and more related to development team rather than product it self. This is an internal attribute of the team which is not directly visible to the customer. Level 4 denotes a more active and mandatory examination of risk and respect to the team who is going to develop the system. The goals of this level are

- Empowered team and rewards
- Project management
- Risk assessment
- No over time
- Simplicity

Level 4 called the improved level includes the people orientation and project management practices. This level focuses on responsibility accepted by the team instead of given to them, considering to do the simplest thing that could possibly works, no hard work but smart work and self organising team.

The AMM at level 4 maturity aims to help developers or managers to respect for the co-workers or people involved in the project, identify and improve problems related to team sustainable pace and organising team by itself. This is achieved by an assessment of current process and to identify where weakness lie. The following figure 3 summaries goals, key process areas and assessment questionnaires for AMM maturity level 4.

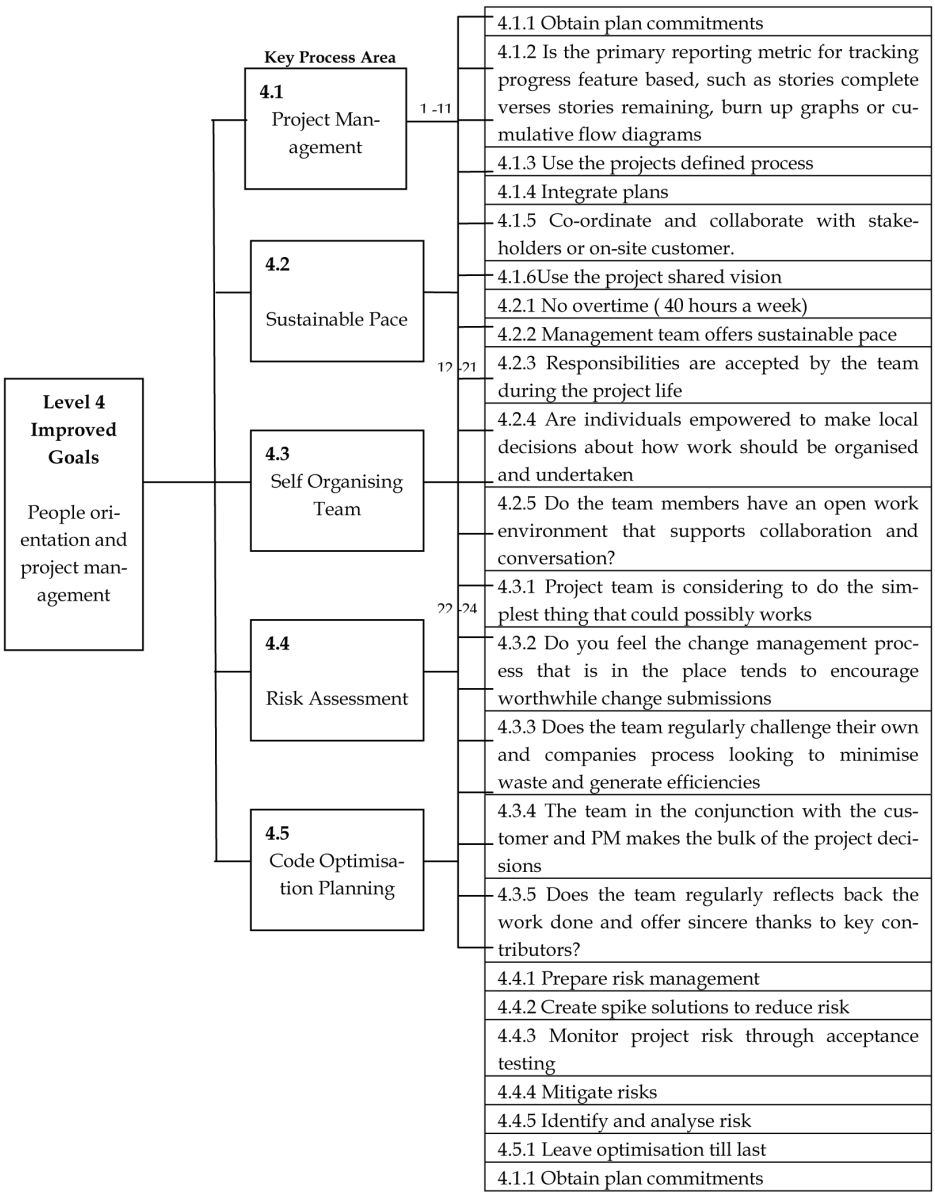

Figure 3 Goal, Key process area and assessment questionnaires for AMM Level 4

Level 5: Mature level (Performance Management and Defect prevention practices)

Companies at this level continually improve their processes through quantitative feedback from the process and form testing innovative ideas and technologies [25]. Companies moving up from level 4 to level 5 should have a wealth of metric data to manage the course of process [11].

The mature level of AMM addresses issues of customer and developer's satisfaction. Here we decided to take into account not only the software process but also result achieved by the team. The goals of this level are

- Context improvement
- Uncertainty management
- Tuning project performance
- Defect Prevention

There are two KPAs at this level which are project performance and defect preventions. The following figure 4 summaries goals, key process areas and assessment questionnaires for AMM maturity level 5.

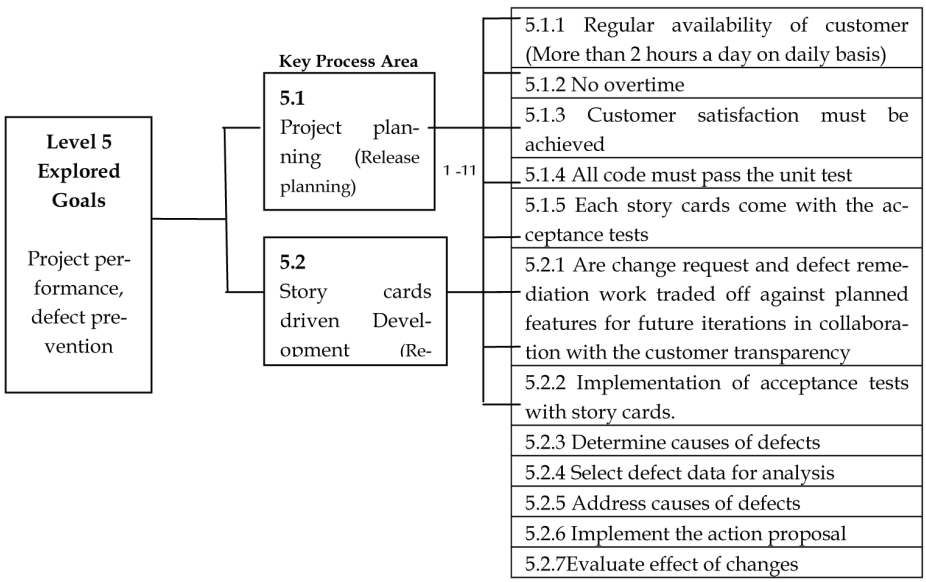

Figure 4 Goal, Key process area and assessment questionnaires for AMM Level 5

3- SOFTWARE PROCESS IMPROVEMENT ROADMAP FOR AGILE SOFTWARE DVELOPMENT PRACTICES

The process improvement roadmap for agile software development is summarised in figure 5.

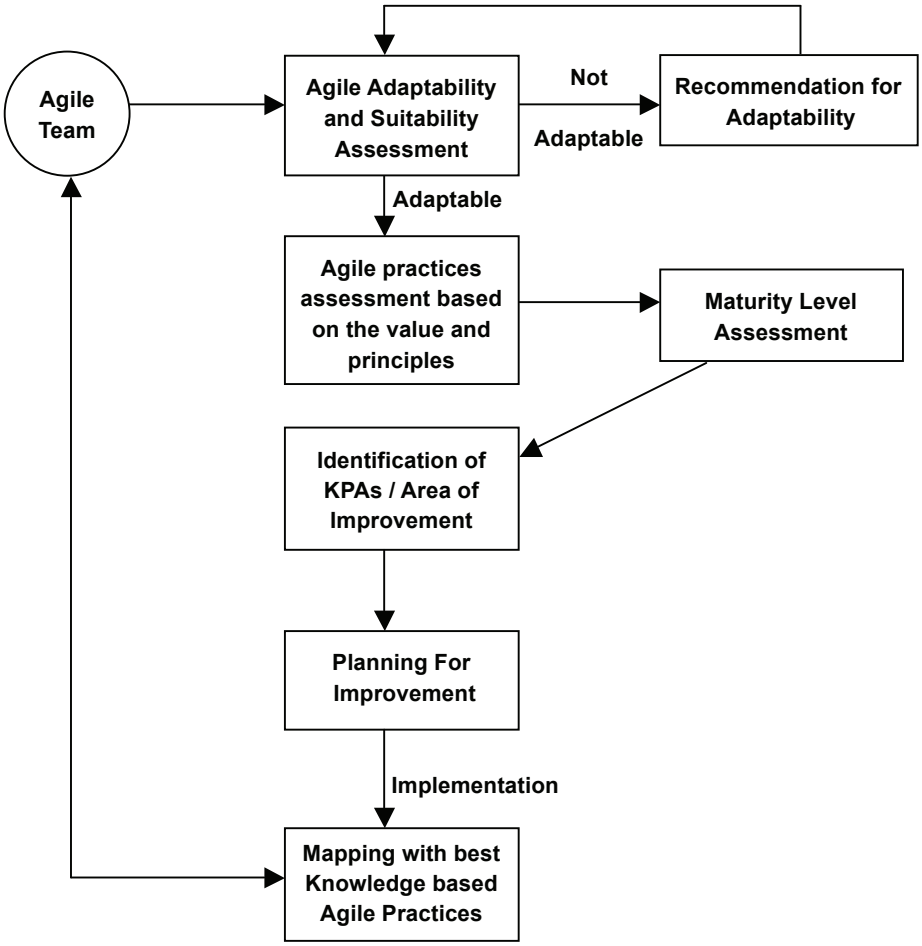

Figure 5 Process improvement roadmap for Agile practices.

The key features of the process are as:

- Adaptability and suitability assessment is carried out by the agile team members which are any like developers, coach, testers with collaboration of on-site customer. This is found to be a useful process during the AMM implementation. The purpose of involving this process is to ensure or to identify the organization is an agile software development organization or not. If not then this adaptability and suitability recom-

mends what they needs to do be an agile software development organization.

- Early in the AMM programme the business objectives or business goal are defined by the agile team. The goals drive much of the subsequent activity, especially the selection of KPAs or maturity level and prioritisation of the area for improvements.
- A tailored version of the AMM assessment (similar like CMMI model but the key process areas and goals are entirely different than CMMI) is carried out by the agile team, to identify area for improvement. This is also indicating the maturity level of the software process.
- The plan for the improvement is identified based on the inputs provided to the assessment questionnaires for each maturity level key process areas. In this plan, practices should be identified to support the implementation of the prioritised area for improvements.
- After the identification of the KPAs for each maturity level, a guide based approach was designed to capture the best practices in order to improve the prioritised area for improvement. This guide based practices approach is found on the agile software development literature like extreme programming practices [3].

3-1 The Adaptability and suitability assessment

Adaptability framework is based on the questionnaires, like the determining the main problems in the existing software development process or software development methodology used or intend to use during the next project, existing knowledge on traditional and agile software development methodology, customer relationship with development team, customer availability during the project, developers attitude or characteristic towards the working patterns, their quality of work individually and in group, project size and lastly manager's attitude towards team and by assessing their knowledge of project by using traditional and agile methods.

An adaptability questionnaire, which is showed in the table2, is actually divided in the following five sections.

- Software development methods used or intend to use.
- Problem identification during the software development and Solution adopted or trying to adopt to solve problems
- Customer availability and relationship
- Developers and Managers knowledge on Agile methods and working quality in group
- Project size (Usually agile methods are suitable for SMEs)

Table 2 Adaptability questionnaires

Adaptability Questions	**Possible options**
Which one are the most difficult software development problems you had or are you trying to solve?	Customer availability Customer relationship Deliver software on time with all features Variable requirements Requirement change Schedule slips Project cancelled System goes sour Defect rate False feature rich Staff turnover Lack of quality staff Excessive documentation of requirements High turn over of employee
Which development life cycle (process) is used or intends to use on pilot project?	Incremental software development Sequential process (Water Fall Mode) Prototyping software development method Evolutionary process Reuse Process
Which factor do you want to optimise during the pilot project?	Productivity Minimum budget Efficient burn rate On-time delivery Beating the estimate Following the plan Working with incomplete knowledge Handling emergent requirements
How often you got customer available at project location?	On-site customer Fixed contract Visit once a week Priced contract Visit when needed Not available at all
What is the primary problem you face with customer or which problems are you trying to solve?	Availability Variable requirements(Different voice) Request to deliver product quickly
Customer's category towards domain	Domain expert Business expertise Business analyst Novice
Which method used to present requirements or going to use on pilot project?	Detailed documentation User story(Story cards) Use case Other

Estimation is done by?	Developer Project manager Project tracker Testing team Quality assurance team
Estimation technique used or planning to use?	Past estimation Function point COCOMO model Other
What do you consider about customer relationship ?	Not satisfied Satisfied Really impressed
Do you do or planning to do up-front project design	Yes or No
What is developers most important working quality	Ability to work in group(Pair programming) High individual ability(Solo programming)
Do your management team consider to do simplest thing that could possibly work?	Yes or No
Do manager offers sustainable pace (40 hours work)	Yes or No
What is manager's view or attitude towards responsibility	Responsibility is assigned or given to team Responsibility accepted by team
How often managers do meeting or ready to do?	Daily stand up meeting Weekly meeting Do when needed Not at all
What is the size of the pilot project?	Small Medium Large

Our adaptability assessment brings three result based on the answers supplied on the adaptability Model. Those results are as following

1. Recommended to adopt agile methodology on you pilot project.
2. Ready to adopt an agile methodology but needs an improvement or needs to pay attention or focus on the recommended area.
3. Pilot project is not suitable for agile methodology, but they can still apply agility after adopting agile software development knowledge

The following figure shows an adaptability framework process. Where developer (Any member of development team) passes through the assessment questionnaires and end of the assessment result is retrieved. These questionnaires require an extensive knowledge of project development life cycle and project software development experience as well. This adaptability just not cover one aspect of the development life cycle it covers all aspect of the software development life cycle and it puts people in the centre of the assessment instead of process itself.

3-2 Agile Practices Assessment method and identification of KPAs for improvement

The purpose of the assessment method is to assess the current agile software development practices. Process assessment consists of the knowledge on agile software development practices and business case workshop, Which focus on process improvement and provides a roadmap for process improvement. The AMM assessment model is based on an agile software development practices, modified and customisable version of the SW-CMM assessment questionnaires. Emphasis placed on the agile practices, developers and on-site customers. This process is expected to enhance the communication and understanding; in particular it is expected to clarify the actual issues of the people involved in the process improvement actions. AMM recommended having a shared vision of the process improvement and any one can control process improvement activities at any stage. The following figure 6 shows how to identify the areas for process improvement. We identify area of improvement through the questionnaires which are discussed into the agile maturity model section.

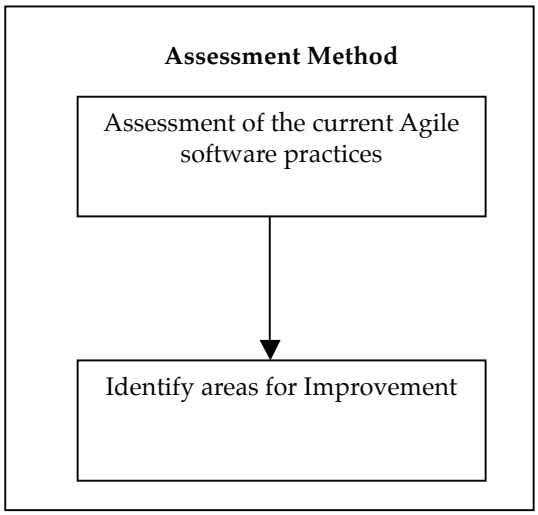

Figure 6 Areas for improvement assessment framework

SW-CMM provides a guideline for good management and engineering practices, with a strong emphasis on management, communication, and coordination for development and maintenance of the software process. But as can be seen earlier SW-CMM does not suitable or acceptable for the agile software development practices or methodology.
The AMM model's main objective is to tailor the software process improvements programme for the agile environments; therefore identifying the maturity level of the agile practices is a crucial activity in the AMM model.

The main objective from the SW-CMM assessment is to assess the capability of an organization and identify the key process area as opportunities for process improvement. The main objective of the AMM assessment is to identify the areas for improvement. This approach is achieved by the AMM trough its own assessment questionnaires based on the agile software development practices, principles and values. In AMM the KPA identifies the issues that must be addressed to achieve a maturity level in AMM maturity model. Each KPA identifies the cluster of goals considered important for enhancing process capability. These related activities are called the key practices. An automated tool has been built to facilitate the work of the AMM method.

The AMM assessment in this project is tailored to suit agile software development environments, their needs and objectives such as eliminating the practices which are not necessary for them and adding new practices which directly related to agile software development. Thus the AMM assessment method is flexible and does not involves any unnecessary KPAs or questionnaires.

Self assessment is the most common way of performing software process assessment [14]. The popularity for self assessment lies in its low cost, good accessibility and ownership of the result [14]. We are going to follow the self assessment for the software process assessment. Automated assessment also considered for this approach.

AMM assessment questionnaires responses are: Yes, Partially, No, Not Applicable (N/A). this assessment response are very similar to SW-CMM response Yes, No, N/A and Don't Know. In our approach response partially permits the assumption that part of the process or work may have been performed or if performed then not fully addressed. N/A is selected when the practice is not possible to implement. If the answer is Yes than the practice is fully implemented and well addressed in the project. If No then it's not addressed at all.

In AMM assessment area of improvement is identified if the answer of the questionnaires is as Partially, No or N/A. Using these criteria the percentage for each KPAs can be calculated as follows:

$$\frac{\sum (Y_n) + \frac{1}{2} \sum (P_n) * 100}{\sum (T_n) - \sum (NA_n)} \qquad (1)$$

Where Yn = Number of Yes answers
Pn = Number of Partially answers
Tn = Total Number of the questions
NAn = Number of N/A answers.

The following table 2 shows the general idea of analysing the questionnaires.

Table 3 General idea of analysing the questionnaires

Answers	No of Answers	Total Questions	Total of Answers Except N/A	KPA rating
Yes	3	7	6	83.33 %
Partially	2			
No	1			
N/A	1			

From table 2 the figure 83.33 in the KPA rating is representing the capability level of the assessed KPA. The interpretation of this as following

- Fully Achieved: 86% to 100% there is evidence of a complete and systematic approach to and full achievement of the defined key practices in the assessed KPA. No significant weaknesses exist across the defined organization unit.
- Largely Achieved: 51% to 85% there is evidence of sound systematic approach to and significant achievement of the defined key practices in the assessed KPA. Performance of the key practices may vary in some areas.
- Partially Achieved: 16% to 50% there is evidence of sound systematic approach to and achievement of the defined key practices in the assessed KPA. Some aspect of achievement may be unpredictable.
- Not Achieved: 51% to 85% there is little or no evidence of achievement of the defined key practices in the assessed KPA.

3-3 Mapping the Area of Improvement with knowledge based Best Agile Practices

Current software process improvement models or CMM models are not compatible or difficult to identify the area of improvement for the agile software development practices. Therefore we suggested using the knowledge of the best agile software development practices that have proven successful in solving problems. Consider the following figure 7, which shows how the identified area of improvements are mapped with the knowledge based best agile software development practices.

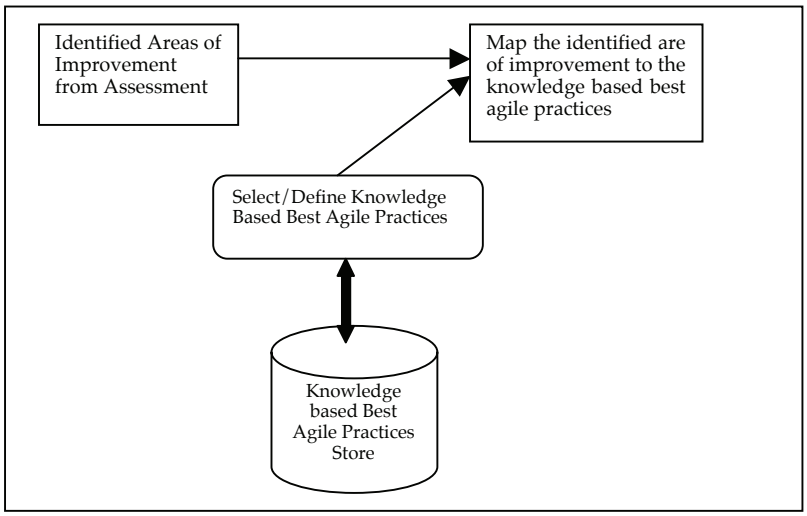

Figure 7 Capturing and mapping are of improvement with Agile best Practices

The figure 7 is the conceptual framework and it is mainly concerned with capturing and enhancing the knowledge of agile software development practices (Agile Practices). The primary concern of the framework is how the process improvement knowledge is captured or identified, how this knowledge is being stored, and how this knowledge of existing agile practices maps to the identified area of improvement. This guide is mainly concerned with the solving particular problems covered during the agile software development practices assessment, and enhances those related agile practices.

Table 4 Example of mapping process improvement best agile practices to area of improvement

KPA	Areas for Improvements	Release Planning	Acceptance Testing	Planning Game	Small Releases	Metaphor	Simple Design	Test Driven Development	Refactoring	Pair Programming	Collective Ownership	Continuous integration	40 hours Week	On-Site Customer	Coding Standard
2.1 Project Planning	2.1.1 The Planning game is used to create project plan.	■	■	■											
	2.1.2 Estimate the Scope of the Project.	■	■	■											
	2.1.3 Release Planning Creates the Schedules.	■		■											

3.4 Mutual Interaction	3.4.1 All code is pair programmed														
	3.4.2 Story cards are written by the collaboration of on-site customer and developers														
	3.4.3 Communicates result through acceptance testing														
4.2 Sustainable pace	4.2.1 No overtime														
	4.2.2 Manager team offers sustainable pace														
	4.2.3 Responsibilities are accepted not given														

In table 3 example cells shaded in grey means corresponds practices in the header of the highlighted cell is mapped to the identified area of improvement within the row of the highlighted cell. That means the identified area of improvement takes the shaded correspondent practices as suggested by the agile team to be suitable to improve the identified area of improvement

We have developed a tool, agile maturity model for measuring the success of Agile software development methodology and also its impact on software process improvement models like CMM (Capability Maturity Model). The purpose of this form is to enable people with little or even no knowledge of agile software development practices, to estimate quickly easily whether agile software development methodology will fulfil their needs and requirements. The program consists of a form containing a handful of simple questions. The answers from these questions will provide immediate feedback on whether agile practices are appropriate for the person who answered the question.

The form will ask questions about the critical areas surrounding agile practices. We need to identify with as few questions as possible whether agile practices are, or are not appropriate. The following aspects have been identified as critical for agile software development practices:

- team size
- client on site
- team location

In order to provide a somewhat more subtle analysis, the following (less critical) aspects have also been selected:

- requirements volatility
- facilities strategy

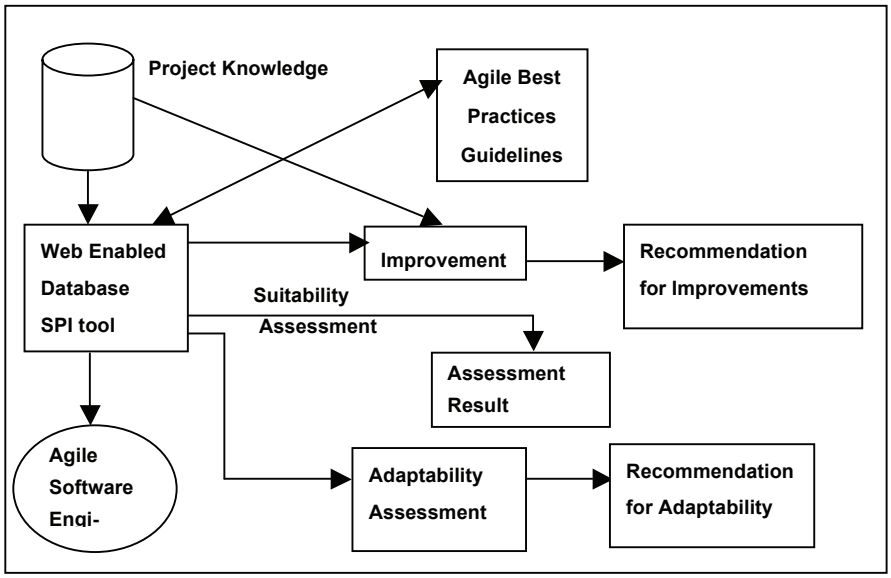

Figure 8 Automated Tool Support

Figure 8 is the illustration of our tool support which provides a web interface and online assessment forms to assess suitability for introducing agile software development practices into any organization. The interface has been made simple thus allowing a first time user to fill in the form right away and getting a result within a few minutes. The results will be colour coded to help result interpretation and a summarised result will also be available. We have developed a web based tool which provides an assessment and analysis for migrating to Agile.

4- RESULTS

We discussed our approach with three different organizations. The following table 3 summarize the participating companies.

Table 3 Participating companies

	Type of company	Business Area	Total number of employees	Number of software developers
Company A	Independent	Flyer Design	28	11
Company B	Independent	Software House	23	9
Company C	Independent	Web development and hosting	19	12

We are still at an early stage of this project, so conclusions are necessarily tentative, and based on informal observation and discussion. All of the technical managers were very supportive of the idea of process improvement framework for agile software development means Agile Maturity Model (AMM) were found in all the companies. Business managers tended to be somewhat more sceptical, and will require evidence of payback before becoming fully convinced of the usefulness of this approach. There was general acceptance and enthusiasm for a more quantitative approach. Company A is now well into the implementation phase of their AMM programme, and already report improvements in project planning and Agile requirements engineering process. However more analysis will be needed to determine if this is in fact a direct result of the improvements initiated as part of the AMM programme. It is important that improvements are applied in key process areas that will provide visible payback within a fairly short period. Certainly there should be measurable benefits visible with about a year from the outset, or else confidence and support for the AMM programme will be eroded. In all companies baseline measurements are being put in place that will allow us to measure the return on investment, and this will be the principal means by which we will evaluate the effectiveness of our approach.

5- CONCLUSION

The capability maturity models for software and process improvement were applicable to the agile methods or not, this is a still challenging issue in the field of the software engineering. In this paper we describe why and how we have adapted the process improvement framework and agile maturity model to focus on agile software development practices. We demonstrate an improvement methodology through a series of models that focus on the adaptability, suitability and improvement process of agile practices. Here we demonstrate how organization can switch into agile organization. In this paper we also developed questionnaires for each level (e.g. figure 2, table 1, figure 3 and figure 4) which will identify the key process area for improvement and which best knowledge based agile practice need to be considered to improve that KPAs by mapping the Area of Improvement with knowledge based Best Agile Practices. This paper will provide a foundation for future development in the area Agile software development process improvement.

6- FUTURE WORK

A validation study of our AMM model is going to carry out with a group of experts in both research and industry. Future work includes creating a more flexible and an automated tool for an assessment to identify the KPAs or area of improvement for agile practices. Verification and evaluation is still required, and future work includes testing the model in an industrial setting.

REFERENCES

[1] Agile Manifesto, (2006) Manifesto for Agile Software Development. [internet], <http://agilemanifesto.org> [Accessed last 01-03-2006]

[2] Ahern, D. M., Clouse, A. and Turner, R. (2003) CMMI Distilled: A Practical Introduction to integrated Process Improvement 2nd ed. UK Addison Wesley.

[3] Beck, K., (2000) Extreme Programming Explained: Embrace Change, Addison- Wesley Press.

[4] Beecham, S., Hall, T. and Rainer, A. (2003), Defining a Requirements Process Improvement Model.. Software Quality Journal Volume 13 Number 13 septembre 2005, Springer, pp.247-279.

[5] Boehm, B. (2003), Value-Based Software Engineering ACM SIGSOFT Software Engineering Notes, 28(2) March 2003, pp 3-15

[6] Boehm, B., Port, D., Jain, A., and Basili, V. (2002), Achieving CMMI Level 5 Improvements with MBASE and the CeBASE Method. [Internet] Cross Talk Journal, Available from : <http://www.stsc.hill.af.mil/CrossTalk/2002/may/boehm.asp> [Accessed 11-10-2007]

[7] Brodman, J. and Johnson, D (1997), A Software Process Improvement Approach for small organisation and small projects. Proceedings of the 19th International Conference in Software Engineering, 19th may 1997, Boston- MA, ACM Press, pp661-662.

[8] Casey, V. and Richardson, I. (2002), A Practical Application of Ideal Model. Product Focused Software Process Improvement, 4th International Conference (PROFES), December 9-11, Rovaniemi – Finland, Springer, pp.172-184.

[9] Chrissis, M. B., Konrad, M. and Shrun, S. (2003) CMMI: Guidelines for Process Integration and Product Improvement, UK, Addison Wesley.

[10] Chrissis, M. B., Wemyss, G., Goldenson, D., Konrad, M., Smith, K. and Svolou, A. (2003) CMMI Interpretive Guidance Project : Preliminary Report [Internet], Software Engineering Institute, Available from :
<http://www.sei.cmu.edu/pub/documents/03.reports/pdf/03sr007-body-revised.pdf> [Accessed 01-06-2004].

[11] Christie, A. M. (1999), Simulation in support of CMM-based process improvement. Journal of Systems and Software, (46): 107-112.

[12] Cockburn, A., (2004) Crystal Clear A Human-Powered Methodology for Small Teams, and it Addison- Wesley Press.

[13] Dangle, K., Larsen, P. and Shaw, M. (2005) software process improvement in small organisation: A case study, IEEE Software November/December 22(6) pp 68-75.

[14] Dutta, S., Lee, M. and Wassenhove, L. K. (1999) Software Engineering in Europe : A study of best practices, IEEE Software Volume (16), Issue(3).

[15] Dyba, T. (2003), Factoes of Software Process Improvement Success in Small and Large Organizations: An Empirical Study in the Scandinavian Context. Proceedings of the 9th European Software Engineering Conference held jointly with 10th ACM ACM SIGSOFT International symposium on foundations of software Engineering, September 2003, Helsinki – Finland, ACM Press, pp.148-157.

[16] Glib, T. (2003), Software Project Management Adding Stakeholder Metrics to Agile Projects. The European Journal for the Informatics Professional, IV(4) August 2003, pp.5-9

[17] Herbsleb, J. D. and Goldenson, D. R. (1996), A Systematic Survey of CMM experience and results. Proceedings of the 18th international conference on Software Engineering, May 1996, Berlin, Germany, IEEE Computer Society, pp.323-330.

[18] Highsmith, J., (2004) Agile Project Management, Creating innovative products, Addison- Wesley.

[19] Ihme, T. and Abrahamsson, P., (2005) The Use of Architectural Patterns in the Agile Software Development of Mobile Applications, International Journal of Agile Manufacturing, Vol. 8, issue 2, 97-112.

[20] Johnson, J., Boucher, K. D., Connors, K. and Robinson, J. (2001), Collaborating on Project Success. [Internet], Software Magazine, Available from : <http://www.softwaremag.com/l.cfm?Doc=archive/2001feb/collaborativeMgt.html> [Accessed 20-jan-2008]

[21] Li, E., Chen, H. and Lee., T. (2002), Software Process Improvement of Top Companies in Taiwan:a comparative study. Total Quality Management 13(5) March 2002, pp701-703.

[22] Lyard, A. and Orci, T. (2000), Dynamic CMM for Small organisations. Proceedings ASSE 2000, the first Argentine Symposium on Software Engineering, September 2000, Tandil- Argentina, pp.133-149.

[23] Nawrocki, J., Walter, B. and Wojciechowski, A. (2001), Towards the maturity model for extreme programming, 27th Euromicro Proceedings 4-6 September 2001, pp.233-239

[24] Paulk, M. C. (2001) Extreme Programming from a CMM Perspective IEEE Software 18(6) November/December 2001, pp 19-26

[25] Paulk, M. C., Weber, C. V., Curtis, B. and Chrissis, M. B. (1995) The Capability Maturity Model for Software : Guidelines for Improving the Software Process (SEI). USA, Addison Wesley.

[26] Paulk, M., Curtis, M., and Weber, C.,(1993) Software Process Maturity Questionnaire : Capability Model version 1.1. [internet], Carnegie Mellon-Software Engineering Institute, Available from <http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.024.html> [Accessed 11-01-2007]

[27] Paulk, M., Weber, C. and Curtis, M.(1999), The Capability Maturity Model for Software. In K. Emam & N. Madhavji (Eds.), Elements of software process Assessment and Improvement. IEEE Computer Society Press, PP.3-22.

[28] Paulk, M.C. (1998), Using the Software CMM in Small Organisations. The Joint 1998 Proceedings of the Pacific Northwest Software Quality Conference and the Eighth International Conference on Software Quality, 13-14 October 1998, Portland, USA, Software Engineering Institute, PP.350-361.

[29] Persse, J. R. (2001) Implementing the Capability Maturity Model, USA, John Wiley & Sons Inc.

[30] Pikkarainen, M. and Mantyniemi, A., (2006) An Approach for Using CMMI in Agile Software Development Assessments: Experiences from Three Case Studies, SPICE 2006 conference, Luxemburg.

[31] Raynus, J. (1999) Software process improvement with CMM, London, Artech House

[32] Schwaber, S. and Beedle, M., (2002) Agile Software Development With Scrum, Prentice Hall.

[33] Tim, K. (2004) Practical insight into the CMMI, USA Artech house

[34] Turner, R. and Jain, A. (2002) Agile meets CMMI: Culture clash or common cause XP/Agile universe 2002, LNCS 2418 pp. 60-69.

[35] Zaharan, S. (1998) Software Process Improvement: Practical Guidelines for Business Success, 1st. USA, Addision-Wesley.