Lean CMMI: An Iterative and Incremental Approach to CMMI-Based Process Improvement

Amr Noaman Abdel-Hamid
Agile Academy
Cairo, Egypt
amr@agileacademy.co

Alaa El-deen Hamouda
Systems and Computers Engineering, Al-Azhar University
Cairo, Egypt
alaa.hamouda@azhar.edu.eg

Abstract— Software Process Improvement (SPI) projects incorporate organization transition risks which may cause many process improvement initiatives to fail. To mitigate these risks, an iterative and incremental approach called 'Process Increments' is used to manage the SPI project.

In this paper, the Configuration Management process area is used as a case study to show the improvement results difference when the 'Process Increments' approach is used. Results are compared with similar projects which didn't use an incremental approach. This approach shifts the focus from adopting new techniques to achieving value-add for the organization and shows excellent results in effective and efficient implementation of Software Configuration Management.

Through our proposed process increment model, we could reach a significant increase in the performance of the software process improvement.

Keywords— Lean, Process Increments, Software Process Improvement, CMMI, Software Configuration Management

I. INTRODUCTION

Recent studies [1, 2 and 3] showed that many Software Process Improvement attempts fail to continue, or fail to show return on investment. Hamouda and El-wahsh [4] noted that one fifth of the companies implementing CMMI-based process improvement initiatives abandoned the standard processes defined. Others complain from "Complexity of operations" and "Process overheads".

In these studies, the SPI projects are driven by process area (Project management, requirements engineering, development, testing, etc.) and these projects focus on one stage at a time when applying process change.

The problem with this approach is that improvement is partial and does not apply to the whole software development process. This postpones realizing value till the end of the project and teams get involved in many upfront activities before seeing any return on their investment.

In this paper, an iterative and incremental approach is suggested. This approach suite the Software Process Improvement (SPI) projects which incorporate organization transition risks, especially when improvement projects involve paradigm shift and organization restructuring towards Agile and lean processes.

II. RELATED WORK

Cohn [5] proposed an iterative model for agile adoption in which teams injects agile practices gradually. He emphasized the importance of iterative adoption but he did not introduce any methodology for defining the contents of every iteration or a roadmap for adopting agile practices.

Sidky [6] proposed an agile adoption framework in which adoption is decomposed into four stages. Each stage defines a set of practices and core concepts and related measurements. Although this framework defines agile practices with each stage, it mentions these practices in a very high level manner. There is no structured description of criteria that team may use to determine whether a practice is fully implemented or not. Also, Sidky [7] discussed the importance of having SCM tools but gave some very high level guidance on the type of practices to be employed.

Rohunen [8] gives an overview of agile adoption methods; results of analysis identified that more recent methods stress on more incremental approaches to adoption. However, increments are generally described in terms of agile practices. No further details are given for describing such practices.

The Experienced-based model [9] for agile adoption starts by modeling the current process and incrementally injecting more practices into this model. It proposes a selection process for prioritizing which practice to adopt first based on analysis of situational factors. Although the model proposes the idea of continuously getting feedback and adjusting the adoption pathway, it has a huge overhead of tools and process definition steps which may not be applicable for small teams.

Other studies [10] [11] focus on finding a best roadmap for practice adoption. Some of them are based on feedback from team or other pre-defined factors. However, in all these studies, there is no defined methodology for defining and prioritizing new practices or techniques other than those defined. Also, they focus primarily on investigating prioritizing agile practice adoption rather than defining a methodology for agile adoption.

In this paper, we introduce the 'Process Increments' methodology, which is an incremental and iterative method for applying process increments. Then, we will analyze data of process improvement and agile adoption projects in 12 companies with specific focus on Configuration Management.

978-1-4673-7153-7/15 $31.00 © 2015 IEEE
DOI 10.1109/Agile.2015.20

III. PROPOSED METHODOLOGY

In this section, we introduce our proposed 'Process Increments' methodology.

A. The 'Process Increments' Method

Process Increments is an iterative and incremental approach to manage improvement projects. The approach builds upon Agile values and principles and reuses some well-known Agile techniques in managing transition risks.

A *process increment* is a process improvement chunk which can be implemented in a relatively small time (1-2 weeks) and still provide value for the organization. A process increment is independent of any other process increment, although it may have prerequisite ones.

The concept of process increment in software process improvement (SPI) projects is almost identical to user stories in Agile development projects.

Fig. 1 describes a map of *process increments*, drawn as rounded rectangles. Each of them may have one or more prerequisite increments, drawn as incoming arrows into the increment. Synchronization bars are used to describe the possibility of parallel application of increments

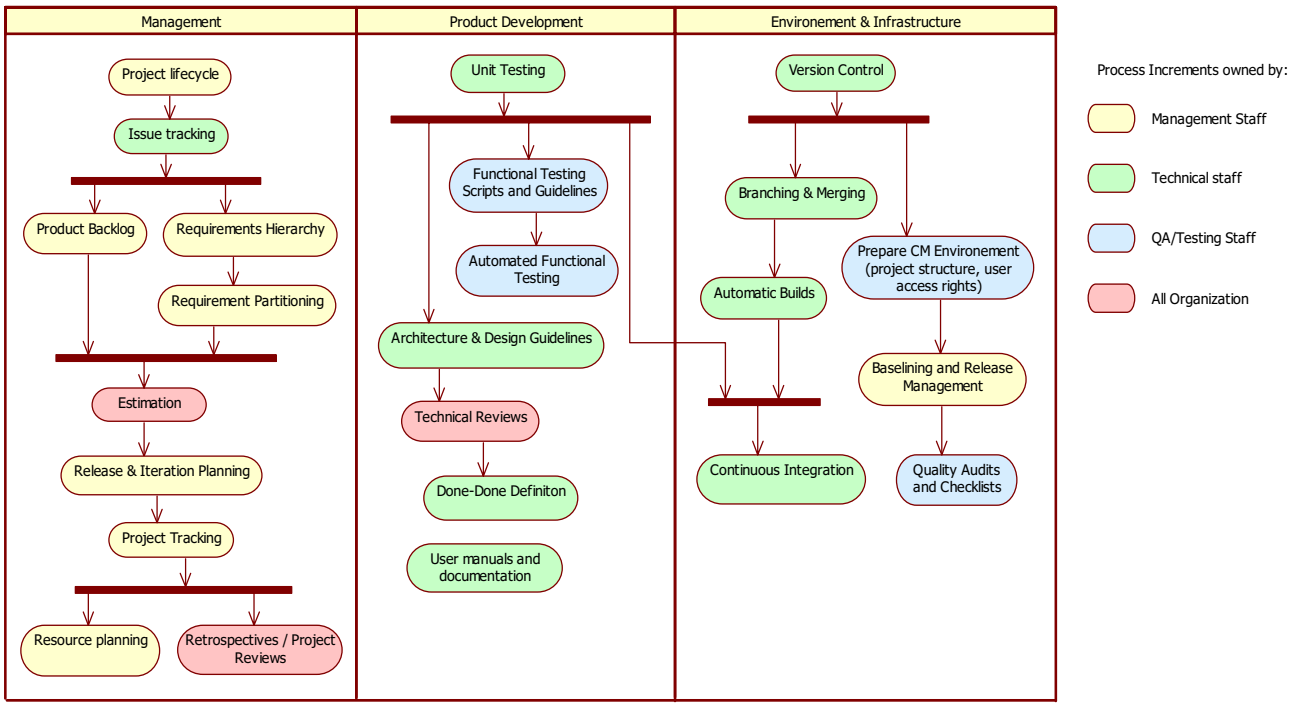

Figure 1. Process increments categories and dependencies

B. Process Increment Attributes

Process increments have the following attributes:

- Summary title: A description of the process increment
- Verification points (aka Conditions of Satisfaction): These are the criteria which when satisfied imply that the process increment is fulfilled
- Size estimate: Similar to a user story, the size of a process increment is estimated in points, using techniques like Planning Poker

Process area: Every process increment is categorized under one of the following broad process areas: (1) Management: includes increments whose main theme is related to management, like estimation, tracking, etc. (2) Product Development: includes increments whose activities mainly focus on developing the product, including testing activities. (3) Environment & Infrastructure: includes increments related to preparing the infrastructure necessary for development

C. Process Increments Lifecycle

Process improvement can introduce major organization changes, and these changes should be managed against risks which may occur. The iterative lifecycle is one of the excellent techniques to manage these risks. Typical improvement iteration is described in the next figure:

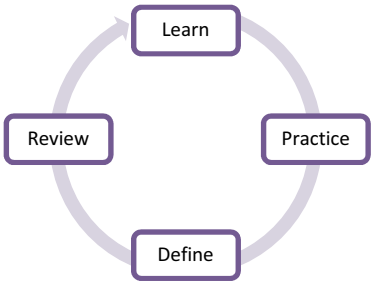

Figure 2. Typical two-week iteration in the improvement project

This is a two-week iteration which contains the following activities:

- **Learn** new concepts. These concepts might be new process increments, or further detailing of a process increment carried over from previous iterations. This learning process is time-boxed to one day.
- **Practice** this new concept and apply it to live projects. This is the major part of the iteration time. The team passes through all the pain of trying new tools, concepts, procedures, etc.
- **Define** what is practiced into guidelines or process documentation. The intent of this activity is to record what the team has learned and practiced to be used while rolling out changes throughout the rest of the organization. Also, most companies are suffering from high rate of turn-over. Developers or testers who participated in this learning experience may leave the company shortly after the program ends. Finally, part of the program is introducing the quality audits into the organization. This is not very common in Agile environments. However, it is recommended when the culture of the working staff is loose and overly relaxed. In this case, high level process guidelines would be excellent criteria for a quality auditor to assess whether the team is implementing what they agreed to be good practices.
- **Review** what has been accomplished with the consultant, and evaluate whether the process increment is done-done or not. The definition of 'done' is presented below.

D. *Agile Practices*

Many agile practices used to manage the SPI project like:

1) *Done Definition*

The done definition is the minimum conditions required so that a process increment is considered completed. In the SPI project, we use this Definition of Done:

1. All verification points are fulfilled
2. The *process increment* is applied to at least one live project
3. The *process increment* is documented in a process guideline, or described in one of the organization process standard documentation

2) *Vertical Slicing of Project Scope*

Project scope is decomposed into user-story like increments. This is a useful technique which gave team the flexibility to implement increments in any order according to the company's priority and business value

3) *Size Estimation and Burn Charts*

Process increments are estimated in size points which indicate the relative weight of every increment. This enabled tracking the whole SPI project using the famous 'Burn up chart' technique.

4) *Iterative and Incremental Delivery*

The project duration is split into timeboxed 2-week iterations. After each iteration, some of the increments are introduced to the team who experiment with it in one live project and documents high level guidelines about it in a wiki page. This resembles the incremental delivery style of agile projects.

IV. PROPOSED METHODOLOGY IMPLEMENTATION

A. *Configuration Management Process Increments*

In this study, we are using Software Configuration Management (SCM) process area as a case study. SCM has been decomposed into Process Increments, and implementation data has been collected and analyzed.

The reason SCM is selected is that SCM is generally perceived to be heavy and hectic to implement, and many research papers are talking about reducing the complexity of CM in different contexts [12]. The main reason for this is the amount of overheads introduced while implementing the practices and techniques related to this process area [13]. A middle-size company would need about 7-10 templates, 4-5 procedures, and many sub-activities to implement configuration management environment. The CM process area is selected to show how incremental methods of process improvement may lead to better results even with such complicated process areas.

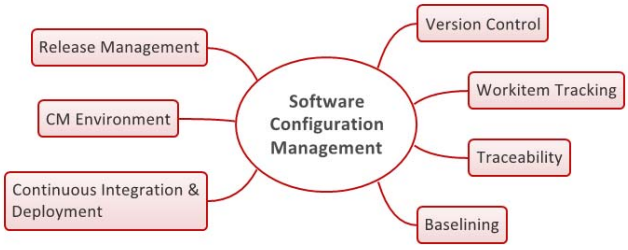

Figure 3. SCM decomposed into process increments

Table I below lists increments fully covering the area of configuration management. It shows how large process areas are partitioned into process increments, and how verification points are written for every increment. It also lists the size of the increment in points, which were group estimated by three experts in the field using the planning poker technique.

In software process improvement projects, this technique was very effective in clearly identifying the scope. It also enabled tracking progress of the SPI project using value-based techniques like burn charts.

TABLE I. SCM PROCESS INCREMENTS DETAILS

Title	Description	Verification Points	Size
Workitem Tracking	Workitem types are identified, and workitems are managed and tracked	• Verify that a tool is selected for issue tracking • Verify that Requirements, Bugs, and CR's are tracked for at least one project • Verify that the issue tracking tool and the version control tool are integrated • Verify that workitem types are identified • Verify that workflows are defined for every workitem type	13
Version Control	Project configuration items are under version control, and team is trained on basic copy-update-merge and lock-modify-unlock procedures	• Verify that VC tool is integrated with the IDE • Verify that team is using check-out check-in (copy-update-merge) procedure to update code • Verify that the code update procedure is documented in the CM guidelines	8
CM Environment	Project structure is defined, access rights are enforced, backup/restore procedures are employed, and proper branching/merging techniques are in-action	• Verify that CM environment is created • Verify that Backup/Restore procedure is defined and experimented successfully • Verify that users are defined and access rights are enforced • Verify that a default project structure is defined in the CM guidelines • Verify that deployment environment(s) is defined, including software/hardware interfaces with deployed packages • Verify that a procedure exists for communicating deployment issues to the development team	13
Release Management	Release and release scope is identified; Changes are received, prioritized, and planned; packaging, releasing, and post-release verification procedures are enforced	• Verify that branching and merging guidelines exist and used for one live project • Verify that versioning scheme is defined and employed • Verify that a packaging, releasing, and post-release verification procedure is defined and employed	8
Baselining	Baselining procedure is defined and enforced at points where work product(s) are delivered to an external party	• Verify that baselines are identified across the project lifecycle • Verify that the baselines contents (work products) are identified • Verify that a specific location for baselines is identified on the CM tool • Verify that team has no update right on the baselines directory • Verify that naming conventions for the baselines are documented and used	5
Traceability	Bi-directional traceability of requirements and work products is defined and enforced	• Verify that requirements are traceable to their design, code modules, and test cases • Verify that requirements are traceable to defects and CR's • Verify that traceability is bi-directional	5
Continuous Integration and Deployment	Builds are automated; and integration between team members, and between teams is automated and frequent; and deployment procedures to customer environments are defined and verified	• Verify that automatic build scripts are developed for at least one live project • Verify that automated tests run to validate generated builds • Verify that the development team are notified by the build status and generated issues (if any) • Verify that a (preferably automated) procedure exists for deploying packages to testing/staging/production environments	20

B. Project Setup

Companies participating in this study has undergone Software Process Improvement program funded by the IT Infrastructure Development Agency (ITIDA) and supervised by the Software Engineering Competence Center (SECC) to help small and medium Enterprises in software process improvement. The program time is fixed to six months and is comprised of three phases:

- **Training:** Companies are trained on some basic principles of software engineering.
- **Consultation & Coaching:** Companies receive guidance and technical support of SECC consultants
- **Evaluation:** Companies are evaluated against 10 CMMI process areas (selected from L2 and L3). The evaluation method is dependent on the SCAMPI method used in formal CMMI appraisal

C. Companies Selection

In this study, the companies' development teams ranged from 7 to 14 persons (counting all roles). Their business domains involve telecom, mobile applications, banking, enterprise applications, and government solutions. They adopt different technology stacks and programming languages like Java, .NET, Delphi, and C++.

V. METHODOLOGY EVALUATION

A. Evaluation Method

The Process Increments methodology has been used in 7 companies to improve their processes. These companies has been subject to SCAMPI A [15] appraisal. SCAMPI A is a formal method to identify *strengths* and *weaknesses* of engineering practices in place. *Weaknesses* are defined as

"gaps in practice implementation" and *strengths* are defined as "exemplary implementation of model practices"[1].

B. Evaluation and Rating of Process Improvement

CMMI Specific Practices are evaluated in two dimensions:

- Process Definition: This amounts for 30% of the score
- Process Implementation in at least one project: This amounts for 70%

Following the SCAMPI method, both process definition and implementation are given one of the four scores:

- Fully Implemented (FI)
- Largely Implemented (LI)
- Partially Implemented (PI)
- Not Implemented (NI)

Then, every score is normalized as a percentage score as follows: FI: 100%, LI: 70%, PI: 40%, and NI: 0%.

Evaluation is done against the Configuration Management Process Area [14], which is composed of the Specific and Generic Practices illustrated in Table II.

TABLE II. CMMI SPECIFIC GOALS AND PRACTICES FOR THE CONFIGURATION MANAGEMENT PROCESS AREA

SG 1	**Baselines of identified work products are established**
SP1.1	Identify the configuration items, components, and related work products that will be placed under configuration management.
SP1.2	Establish and maintain a configuration management and change management system for controlling work products.
SP1.3	Create or release baselines for internal use and for delivery to the customer.
SG 2	**Changes to the work products under configuration management are tracked and controlled**
SP2.1	Changes to the work products under configuration management are tracked and controlled.
SP2.2	Control changes to the configuration items.
SG 3	**Integrity of baselines is established and maintained.**
SP3.1	Establish and maintain records describing configuration items.
SP3.2	Perform configuration audits to maintain integrity of the configuration baselines.

C. Results Before and After

To evaluate methodology effectiveness, CMMI SP scores for these 7 companies which used the Process Increments method have been compared to scores of 5 companies which did not use the Process Increments approach.

Table III and Table VI list the number of weaknesses in the CM process area and the average company rating for all SP scores of the CM process area. The first table lists results for companies which did not follow the 'Process Increments' method, whereas the second table lists results for companies which followed the 'Process Increments' method.

[1] Reference [15], page 33.

TABLE III. SPI RESTULS BEFORE 'PROCESS INCREMENTS'

Companies	Count of Weaknesses in CM SP's	Average Company Rating for SP's of the CM Process Area
Company 1	5	54%
Company 2	5	9%
Company 3	4	58%
Company 4	4	30%
Company 5	1	90%

TABLE IV. SPI RESULTS AFTER 'PROCESS INCREMENTS'

Companies	Count of Weaknesses in CM SP's	Average Company Rating for SP's of the CM Process Area
Company 6	1	96%
Company 7	1	86%
Company 8	4	71%
Company 9	5	94%
Company 10	3	69%
Company 11	3	76%
Company 12	2	75%

Given that all companies are similar in size, program objectives are common among all companies, and period of implementation is fixed to six months, given these three conditions, we can compare results across organizations and these results reliably indicate our process improvement methodology effectiveness.

Figure 4 shows a box plot of CMMI Specific Practice (SP) ratings for companies before and after using the 'Process Increments' methodology. It shows significant improvement in the effectiveness of the SPI project are realized by using an incremental and iterative approach like 'Process Increments'

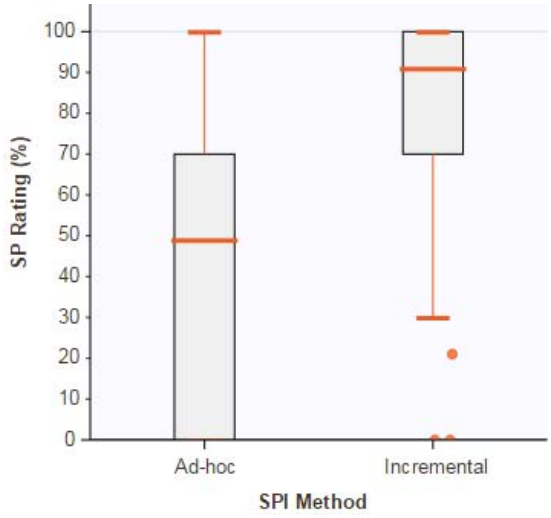

Figure 4. Box-plot of measured improvement

D. Analysis of Weaknesses of Process Implementation

The total number of weaknesses reported for the first group of companies (not using the Increments method) is 19 weaknesses and for the second group (suing Process Increments) is also 19 weaknesses. Table V presents some measurements about the weaknesses before and after:

TABLE V. WEAKNESSES ANALYSIS BEFORE AND AFTER USING 'PROCESS INCREMENTS'

	Before	After
Percentage of "Not Implemented (NI)" weaknesses	14/19 = 74%	2/19 = 11%
Percentage of Weaknesses related to process documentation	0%	6/19 = 32%

The percentage of NI weaknesses was very high and it dropped dramatically from 74% to 11% after using the 'Process Increments' methodology.

This indicates that although the number of weaknesses is the same for both groups of companies, the severity of findings is dramatically reduced after using the Process Increments method. Moreover, in the second group, 32% of weaknesses are related to simple process documentation issues and not to the CM related development activities. Such issues are very light and easy to fix.

CONCLUSION

SPI projects are risky and may easily fail if these risks are not properly mitigated. After applying our proposed 'Process Increments' methodology, CMMI-based process improvement projects were much more successful. The SPI results have dramatically improved. The main reason behind that is the agile nature of the 'Process Increments' method.

Several key factors are behind the success of the 'Process Increments'. One of them is decomposing the project scope into small valuable slices of organizational changes, then sizing and prioritizing these changes. This enabled responding to changes in organization's priorities and risks. Also, another key factor was the higher visibility of the scope and progress of the SPI project through continuous planning and tracking using iterations and burn charts.

REFERENCES

[1] Ahmed Roshdy El Shalaby, "Capability Maturity Model (CMMI) Adoption by Egyptian Software Companies: Motivations and Barriers", MBA thesis, May, 2009.

[2] Alaa Hamouda, "Using Agile Story Points as an Estimation Technique in CMMI Organizations", Agile Conference, 2014. (IEEE DOI 10.1109/AGILE.2014.11).

[3] Amr Noaman and Mohamed Amr Abdel-Kader, "Process Increments:An Agile Approach to Software Process Improvement", Agile Conference, 2011.

[4] Alaa Hamouda and Mohamed El-wahsh, "Comparative Study of Software Engineering Processes in Egyptian CMMI Companies", The Journal of American Science, ISSN 1545-1003, Volume 6, Issue 11, Cumulated No. 32, November 1, 2010.

[5] Mike Cohn, *Agile Estimating and Planning*, Prentice Hall, 2005.

[6] A. Sidky, J. Arthur, S. Bohner, "A Disciplined Approach to Adopting Agile Practices: The Agile Adoption Framework". Innov Syst Softw Eng 3: pp. 203-216, 2007

[7] A. Sidky, "A Structured Approach to Adopting Agile Practices: The Agile Adoption Framework", Doctoroal Dissertation, May 2007.

[8] A. Rohunen, P. Rodriguez, P. Kuvaja, L. Krzanik, J. Markkula, "Approaches to Agile Adoption in Large Settings: A Comparison of the Results from a Literature Analysis and an Industrial Inventory", Lecture Notes in Computer Science Volume 6156, 2010, pp 77-91.

[9] I. Krasteva, S. Ilieva, A. Dimov, "Experience-Based Approach for Adoption of Agile Practices in Software Development Projects", Lecture Notes in Computer Science Volume 6051, 2010, pp 266-280.

[10] A. Solinski, K. Petersen "Prioritizing agile benefits and limitations in relation to practice usage", Software Quality Journal, September 2014, pp 1-36.

[11] M. Senapathi "A Framework for Situated Evaluation of Methodology Usage in Agile Environments", Lecture Notes in Business Information Processing Volume 48, 2010, pp 407-409.

[12] K. Kannan, N. C. Narendra, and L. Ramaswamy, "Managing Configuration Complexity during Deployment and Maintenance of SOA Solutions", IEEE International Conference on Services Computing, 2009, pp. 152-159.

[13] Usman Durrani and et al, "Lean Configuration Management Systems implementation for the governance: A cloud computing case study", Journal of information technology management, Volume XXV, Number 1, 2014.

[14] "CMMI® for Development Model, Version 1.3", CMMI® Institute, November, 2010

[15] "Standard CMMI® Appraisal Method for Process Improvement (SCAMPI℠) A, Version 1.3: Method Definition Document", Software Engineering Institute (SEI), Carnegie Mellon University, March, 2011