Research Journal of Management Sciences ISSN 2319–1171
Vol. **4(1),** 8-14, January **(2015)** *Res. J. Management Sci.*

Agile Software Development Dynamics through TACIT Knowledge: An Attempt to Acquire and Share TACIT Knowledge

Uma Sankar S.S.[1] and R. Jubi[2]
[1]Research and Development Centre, Bharathiar University, Coimbatore, Tamilnadu, INDIA
[2]Mar Thoma Institute of Information Technology, Kollam, Kerala, INDIA

Available online at: www.isca.in, www.isca.me
Received 10th December 2014, revised 29th December 2014, accepted 5th January 2015

Abstract

Knowledge Management is an emerging discipline that targets to capitalize organizations' intellectual capital. Software development requires intensive sharing of knowledge. Explicit and Tacit knowledge acquired during the project execution shall be captured and managed to meet organizations' objectives. Creating and utilizing knowledge is now becoming the key factor to sustain and obtain competitive advantage. Agile Software development processes extensively uses Tacit knowledge to minimize transaction costs of software engineering activities. The way of eliciting such knowledge to the rest of the team and procuring it as an intellectual asset to the organization is still unknown. This study is an attempt to propose the process for eliciting Tacit knowledge by converting it into Explicit knowledge thereby sharing it across the organization to create intellectual knowledge base to build competitive edge. The proposed concept is being piloting in one of the Agile development project 'Study Management Plug-in' (actual name withheld) in Ultrasound modality software application.

Keywords: Knowledge management, Tacit knowledge, agile software development, SCRUM, SECI Model.

Introduction

Knowledge Management is an emerging discipline that targets to capitalize organizations intellectual capital. It is a set of processes to create, store, transfer, and apply knowledge. Explicit and Tacit knowledge has been acquired during the business processing. The administration and management of such knowledge facilitates the organizations to create strategic asset and leverage decision making processes. Such unique knowledge generated during the business processing is a key contributor to build market competitiveness.

Software development requires intensive sharing of knowledge. Project teams acquire Explicit and Tacit knowledge as part of their involvement in the projects. Explicit knowledge can be articulate, codified, stored, and transferred through documents[1]. Tacit knowledge is evolving as a by-product of the development process. Tacit knowledge is difficult to articulate and measure[1]. However, use of Tacit knowledge is an essential factor to build high-trust development environment.

Agile software development is a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams[2]. Agile Software development processes extensively use tacit knowledge to minimize transaction costs of software engineering activities by maintaining task-mature, well-trained, cohesive and high-trust small teams. Tacit knowledge sharing takes place face-to-face during Agile Scrum meetings. The way of eliciting such knowledge to the rest of the team and procuring it as an intellectual asset to the organization is still unknown. The following sections examine the way of acquiring tacit knowledge and try to quantify its impact in projects.

Knowledge: an Intellectual Organizational Asset

Knowledge is one of the key determinants to meet organization's objectives. It is generated as part of business processing (Product and Services). It is in Explicit and Tacit form. Knowledge is an intellectual asset, because it is unique, generated as part of the organizational activities, utilizing to build competencies and can assert ownership. It acts as a catalyst to leverage the process to meet organization's objectives. Organization uses this asset to derive corporate strategies, short team goals and day to day decision making.

Knowledge management in Software development: Demand on Software solutions to diverse industrial sectors is increasing exponentially. Shorter 'Time to Market' (TTM) with intended Quality and Productivity is one of the challenging factors that software companies are facing today. Since Information Technology is emerging into an industrial sector, there is still lot of limitations to adapt best practices to meet customer expectations. Major cause of such limitations compared to other industrial sectors is, Software development requires intensive sharing of knowledge. Knowledge is stored in paper (Explicit) and in people's mind (Tacit). Explicit knowledge has limited accessibility and become obsolete in fast growing global market. Tacit knowledge is lost when an individual leaves the organization. Attrition rate is very high in Software industry. One in four employees in the organized sector in India is set to switch

jobs, the highest attrition rate globally, according to a Hay Group study[11]. To avoid such limitations and losses, Software organizations need to establish a culture by which knowledge has to be systematically collected, stored in a corporate memory, and shared across the organization.

Tacit knowledge capture

The most difficult form of knowledge is Tacit knowledge. It is intangible and difficult to identify and capture. It is gained through experience and expertise over time in association with day-to-day activities of an individual and resides inside his head. The way of eliciting this intellectual asset from an expert's head is an art rather than a specific well planned process, because Tacit knowledge possesses the following characteristics: i. Tacit knowledge is intangible and resides inside the expert's head as subconscious thoughts. ii. The person may not be aware that he posses valuable knowledge that needs to be shared. iii. Expert knows his knowledge but may not be willing to share. iv. Difficulty in identifying experts since it is contextual. v. Limitations to express knowledge in oral and written forms. vi. Limitations to notice or grasp such knowledge through actions. vii. Applications of tacit knowledge is context specific, educating such specialties has limitations.

Methods for eliciting tacit knowledge

Brain Storming: Conduct Brain Storming for specific problems (technical/non-technical) and document the final conclusions.

Tips and Experience: Share Tips and Experience in Scrum meetings.

Dos and Don'ts: After code and design reviews, reviewers shall share the best practices and things to be avoided as feedback.

Quality Circles: Quality Circles (QC) facilitate the team to evaluate the efficiency of the current process and check the possibilities of further improvements. QC is widely practiced process in Industry to evaluate the current process and take appropriate actions by involving the whole team[5].

Community of Practice (CoP): Communities of practice are groups of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise in this area by interacting on an ongoing basis[3].

In-plant training: Associating team to experts in work, specifically associate junior members to the experts in the team shall facilitate the junior team members to understand expert's approach. Such knowledge would be untold and it is Tacit.

Nonaka's SECI Model

Ikujiro Nonaka and Hirotaka Takeuchi propose a model of the knowledge creating process. The creation of knowledge is a continuous process of dynamic interaction between Tacit and Explicit knowledge[6]. The four modes of knowledge conversion interacted in the spiral of knowledge creation is represented in figure-1.

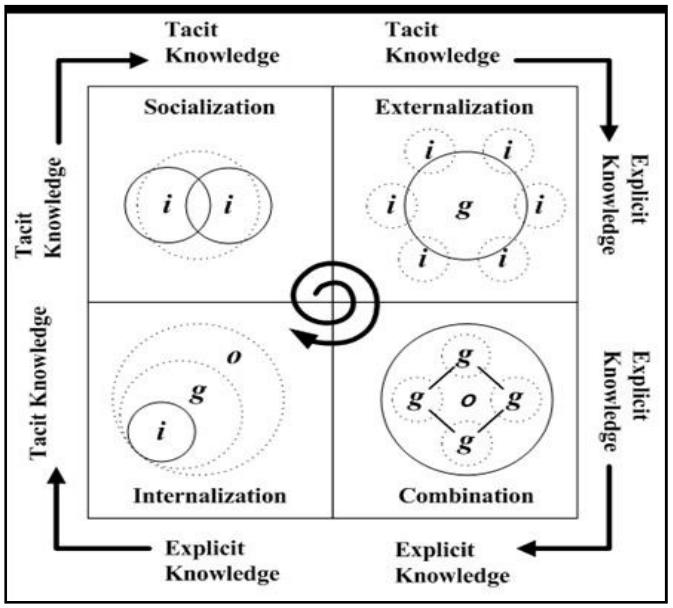

Figure-1
Nonaka's SECI Model

Socialization: Sharing tacit knowledge through face-to-face communication or shared experience.

Externalization: Developing concepts, which embed the combined tacit knowledge and which enables its communication.

Combination: Communication of various elements of explicit knowledge.

Internalization: Closely linked to learning by doing, the explicit knowledge becomes part of the individual's knowledge base and becomes an asset for the organization

Related studies in Tacit knowledge management

Knowledge management is one of the key research topics in the current competitive era. Industries adopt and practice appropriate techniques to preserve and share the knowledge to build their intellectual asset.

Literature review revealed remarkable contributions that have been made by Michael Polanyi who may be credited as 'father' of Tacit knowledge. Ikujiro Nonaka and Hirotaka Takeuchi who derived SECI model to represent the Tacit to Explicit knowledge conversion.

Many studies related to Tacit Knowledge capture in different sectors of industries are found in internet. The topic which is

relevant to this study is done by Peter Anthony Busch, on his thesis titled 'Knowledge Management Implications of Articulable Tacit Knowledge: Case Studies on its Diffusion'. It focuses on Tacit knowledge diffusion within an IT domain. One of the future research problem (scope) highlighted in the said thesis is that studies can be further performed on an organization on the basis that it is suitable to their working environment to capture Tacit knowledge.

So, this study focuses the way to capture Tacit knowledge in an Agile software development model and tries to evaluate its impact in software delivery.

Agile Software Development Dynamics and Tacit Knowledge

Quality, Cost and Time are uncertain for Software projects. Uniqueness and adaption to frequent changes are its inheriting nature. Industries therefore seek best practices and models to adopt for better predictable results in software project management.

Agile Software development facilitates highly interactive, iterative and incremental workflow. According to Bohem "Agile methods are very lightweight processes that employ short iterative cycles, actively involve users to establish, prioritize, and verify requirements, and rely on a team's tacit knowledge as opposed to documentation"[7]. Agile Software development focuses on sharing knowledge among the stakeholders through face-to-face conversation. This facilitates to avoid traditional heavy weight document centric process practices. Since it is face-to-face and real time, its benefits are limited to those who are involved in these conversations. Major part of such knowledge come from the individual member's experience and skill, such knowledge is Tacit.

Agile Software development process is closely linked with Nonaka's SECI model. Socialization happens in daily Scrum meeting, where the team members share their experience and tips. Such knowledge may be untold and might be evolved only through face-to-face discussions. The team member gets this Tacit knowledge and stored in their head. Combination happens when Explicit knowledge such as cross functional and review inputs shared during the Scrum meetings. When Scrum master document the relevant points it becomes Externalization mode of SECI. As a collective result, the Tacit knowledge captured during Scrum meeting and involvement of those team members in other organization activities enables Internalization.

So, the dynamics of Agile Software development can be enriched by facilitating the Tacit knowledge sharing among the team and make it Explicit to organization's intellectual asset.

The forthcoming sections examine the Tacit knowledge captured in 'Study Management Plug-in' project by adopting the Tacit knowledge eliciting methods explained in Section 3.1.

Application of the proposed concept in 'Study Management Plug-in' project

Literature review done in this scope of study provides a substantial evidence to believe that there is an influence of one's intangible knowledge (Tacit) to his output and such shared knowledge has a positive impact to the whole team. This case study examines the validity and applicability of the hypothesis in 'Study Management Plug-in' project which follows Agile development methodology.

'Study Management Plug-in' is one of the core modules in Ultrasound modality software product. It facilitates the Sonographer to manage scan examination. The team is consisting of 12 software engineers with years of experience ranges from 3 to 8 years. Scrum meeting is conducted daily to track the status, decide plans for the day, share experience, share feedback and tips. Agile Manifesto says "Individuals and interactions over processes and tools" [9]. So, one of the primary objectives of Scrum meeting is to provide an environment by which the interaction between cross functional team is taking place effectively. The ultimate aim of such interaction is to elicit the Tacit knowledge from the respective team members.

Tacit knowledge elicitation in 'Study Management Plug-in': Project team is following most of the Tacit knowledge elicitation methods described in section 3.1.

'Brain storming' is conducted on demand to finalize the functional specification and design. Experts participate in such brain storming sessions and share their ideas as well. The respective team members present the design proposals to this forum and acquire team inputs. This facilitates the respective team member(s) to confidently present and proceed with the proposal. Experts evaluate its merits and demerits and conclude the acceptable design. As a bi-product, whole team gets a chance to share their ideas and it also enables the team members to understand the other module's specification and design.

'Tips and Experience' sharing is one of the basic agenda in daily scrum meeting. Tips shared in such scrum meeting were evaluated and rewarded. Review feedbacks and relevant experience sharing also takes place in scrum meetings. 'Do's and Don'ts' sharing is also part of this review feedback sessions.

'Quality Circles' are practiced after every Sprint releases. The focus is to evaluate the improvement points, best practices that have to be continued and action plans to overcome the problems faced. The whole team involves in the brainstorming sessions. The details are documented in KEPUTO format.

'Community of Practice' (CoP) has been practiced in this team by interacting with senior members in core functional areas. They share relevant inputs applicable to 'Study Management Plug-in' module in Design, Code and Process improvement areas.

'In-Plant training' is effectively being practiced in this project. This is achieved by inviting team members in specification, design and review feedback sessions. The junior members are assigned with senior members to provide the orientation to agile practices, system overview and day-to-day technical and non technical activities.

Implications of Tacit Knowledge share in 'Study Management Plug-in' project

Defect is the most important determinant to assess product quality. Defect Density (Defects caught in code reviews and defects caught in developer testing (Defect/KLOC)) has been taken here as an input to evaluate the impact of Tacit knowledge in project.

Table-1
SECI Process output (Tacit to Explicit)

Information
Debugging efficiency: If any component shows functionality error, then verify the other integrated module code implementation which is correctly working
Code Quality: CppUnit facilitates the team to trace and find minute issues
Knowledge sharing: Reviews facilitate the knowledge transfer from the reviewer to the team members (Reviewee)
Code Quality: Scope of Review can be extended to provide better and optimized design/coding tips
Positive Reinforcement: Team members get confidence and satisfaction, when their idea or suggestions got implemented
Team building: Good relations improve sharing ideas and improve confidence
Design know how: Design presentations during the middle of project(was called design marathon) helped to understand the design of overall product
Preventive measures: Daily scrum is helpful to discuss all issues faced in project and able to synch it with other team mates
Knowledge sharing: Tips sharing during Scrum meeting will enhance knowledge sharing among the team
Review effectiveness: Review checklist facilitate to standardize the review. Reviewer can do review systematically and uniformly based on the checklist points
Preventive measures: Important review comments are discussed among the reviewers and team members so that it can be taken care while coding/reviewing other files also.
Productivity: Cpplint is running before sending the files for review, the coding standards errors can be avoided during this step itself review. It increases review efficiency, review speed and code quality.
Design Quality: As design is discussed among team members, most issues can be foreseen and rectified before coding or testing itself. It helps to avoid last time bug fixing.
Optimum Schedule Pressure: As volume of task planned is optimum, we can complete coding, review and rework one day before release and we can have a tension free on-time delivery.
Code Quality: Using cppunit, developers can track and solve the issues in each function at the coding time itself. It reduces testing and bug fixing efforts thereby rework efforts.
Cross functional team integration: Through daily scrum meeting everyone knows the tasks for the sprint and status of each task. So the task assigned to one member can be easily take up by someone else in the team.
Sprint Planning: For 10 days Sprint, tasks is planned only for 7–8 days. This is to address any critical issues during integration
Team building: Responsibility of feature update in different modules are rotated in order to make everyone aware of all modules.
Brain storming: Brain storming done before every critical modification, to get suggestions from team members
Multilevel Review: Designer presents the design before all team members and it will be reviewed during presentation itself. It facilitate combined review.
Code Quality: Function decomposing enables better readability, maintainability and improve code quality
Productivity: Design patterns enables effective review in addition to reuse
Self Review: Review checklist is prepared for self review.

SECI in 'Study Management Plug-in': Sample of information captured during the scrum meetings, technical review meeting and KEPUTO analysis are shown in table-1. Such knowledge might be a matter of common sense or not noticed as a positive input. Some information might be untold facts and very specific to our working environments. By definition, such knowledge is Tacit.

Impact of Tacit knowledge in 'Study Management Plug-in': Defects caught in 6 Sprints during August 2013 to December 2013 are taken here for analysis. The scope of analysis is limited to pre shipment defects because, the integration of this plug-in to the product is yet to be done and therefore no acceptance verification has happened at customer end, so post shipment defects are unavailable.

Consolidation of defects has been done using the code review report for last 6 sprints (August to December 13). The defects were mapped to respective defect causes. QMS review template has been used for code review. Pareto analysis with 80-20 rule applied to filter out the prominent causes which contribute the major part of defects (80% of defect is due to 20% of causes). So, this analysis focuses on the impact of Tacit knowledge to reduce these defects.

Figure-2 and figure-3 shows that the 80% of defects is due to causes such as Coherence (23.9%), Optimization (21.0%), Logic (15.8%) and Functionality (15.2%). These causes are collectively named as 'Core Causes'.

Tacit knowledge sharing was happening in 'Study Management Plug-in' module from the beginning of the project itself, where daily scrum meeting includes the sharing of review feedback and technical tips. But, Tacit knowledge share and make it Explicit across the team started from middle of August 2013 (Sprint 27 onwards).

Figure-4 shows the trend of review defect density for the core causes identified during last five months. Refer table-2 for Defect Density (Defects/KLOC) form Sprint 27 to Sprint 32.

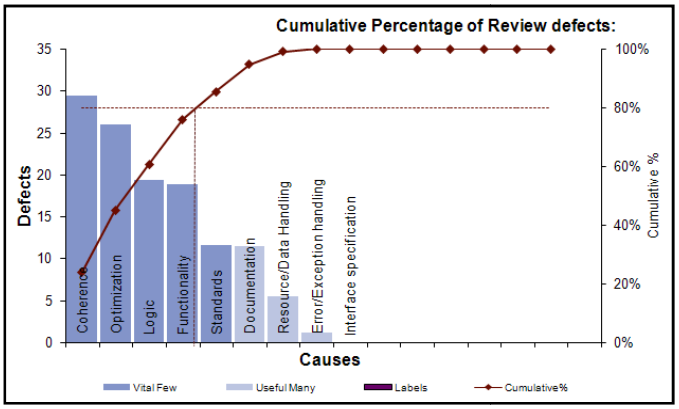

Figure-2
Defect Intensity (80-20 Rule)

		Cumulative Percentage Cutoff:	80%
#	Causes	Defects	Cumulative%
1	Coherence	29.43	23.9%
2	Optimization	25.96	44.9%
3	Logic	19.41	60.7%
4	Functionality	18.83	75.9%
5	Standards	11.61	85.3%
6	Documentation	11.45	94.6%
7	Resource/Data Handling	5.50	99.1%
8	Error/Exception handling	1.12	100.0%
9	Interface specification	0.00	100.0%

Figure-3
Cumulative defects

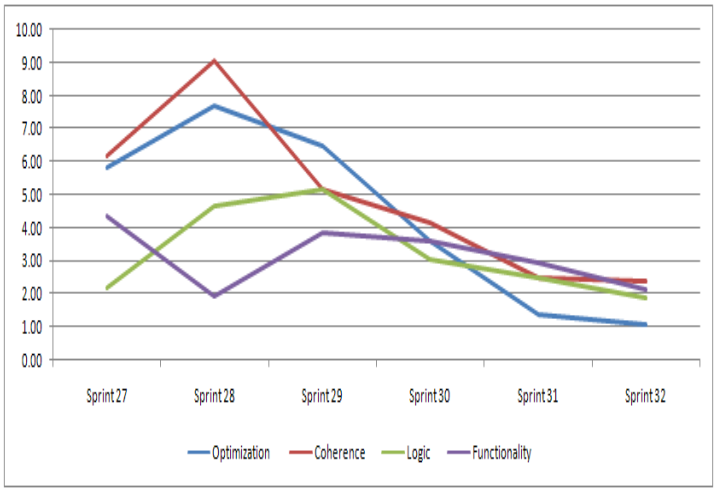

Figure-4
Review defect density trend

Table-2
Defect Density Sprint wise

Sprint	Sprint 27	Sprint 28	Sprint 29	Sprint 30	Sprint 31	Sprint 32
LOC / Defect Type	2754	3651	3876	3601	4425	3741
Optimization	5.81	7.67	6.45	3.61	1.36	1.07
Coherence	6.17	9.04	5.16	4.17	2.49	2.41
Logic	2.18	4.66	5.16	3.05	2.49	1.87
Functionality	4.36	1.92	3.87	3.61	2.94	2.14
DD TOTAL /Sprint	18.52	23.28	20.64	14.44	9.27	7.48

Figure-5 shows the trend of DT defect density caught for the last five months. Refer Table-3 for DT defect density (Defects/KLOC) form Sprint 27 to Sprint 32.

Table-3
DT Defect density Sprint-wise

Sprint	Sprint 27	Sprint 28	Sprint 29	Sprint 30	Sprint 31	Sprint 32
LOC	2754	3651	3876	3601	4425	3741
DT bugs	6	7	7	6	7	0
Bugs/KLOC	2.18	1.92	1.81	1.67	1.58	0.00

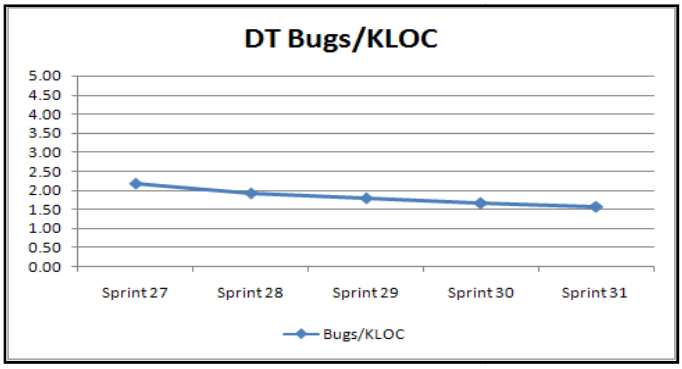

Figure-5
DT defect density trend

The negative slope in the trend graphs (Review defect density (figure-4) and DT defect density (figure-5)) shows that there is a significant improvement in quality after practicing this process. In Sprint 28 some of the core causes has been increased. This is because, as part of the Tacit knowledge share initiative, review checklist has been introduced to strengthen the review.

Statistical analysis and Interpretation: Statistical analysis is essential to ascertain the validity of the findings made out of samples. In statistical hypothesis, NULL Hypothesis (H_0) stands for general or default state. Alternative Hypothesis (H_1) stands for any states other than H_0. In such testing either accept or reject H_0 based on statistical significance (Statistical test).

Two samples with 6 set of sprint data has been taken for testing the hypothesis. First set represents the defect density of code review from Sprint 21 to 26 (table-4). In this case Tacit knowledge capturing is not planed. The second set contains the defect density of code review from Sprint 27 to 32 (table-2) where Tacit Knowledge is captured and shared.

Test statistics: H0: Two samples are same (Mean is same), H1: Two samples are different (Mean is different)

'Two Sample T test' (using MINITAB tool) has been used to validate, whether there is a significant variance in 'Mean' (it means samples are different). Refer the result in Figure-6. If P-Value (probability of obtaining a test statistic) is ≥ 0.05 then accept H_0 otherwise reject H_0, i.e. accept H_1. The result shows that P-Value = 0.038 which is less than 0.05. So, H_0 has to be rejected. It means that the two samples are different. Next, examine whether there is a significant improvement in quality, i.e., check whether which sample has less (defect density) 'Mean'. Refer Figure-6. The mean of sample one is 25.31 and sample two is 15.60.

So, the statistical inference is, there is a significant improvement in quality after practicing Tacit knowledge capture and share in 'Study Management Plug-in' project.

Table-4
Defect Density Sprint 21 to 26

Sprint	Sprint 21	Sprint 22	Sprint 23	Sprint 24	Sprint 25	Sprint 26
LOC / Defect Type	2980	2986	2085	2808	1563	1326
Optimization	4.03	8.71	14.39	4.99	8.32	2.26
Coherence	3.02	4.35	10.07	6.05	6.40	9.80
Logic	3.36	7.70	5.76	5.70	7.68	6.79
Functionality	3.69	6.70	5.76	3.92	6.40	6.03
DD TOTAL /Sprint	14.09	27.46	35.97	20.66	28.79	24.89

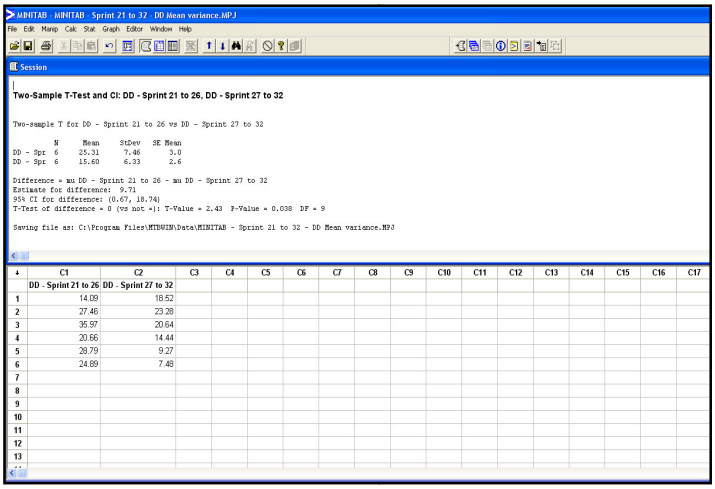

Figure-6
Two Sample T test - Result

Conclusion

Knowledge sharing is happening in the team as an interactive process. Explicit knowledge is stored and available for future reference. Tacit knowledge is stored in individual's head and losses when they leave the job. Organizations loose valuable assets if they fail to capture and maintain it as an intellectual asset to build its competency.

Literature review and application of Tacit knowledge captured in our projects provide significant evidence to prove that Tacit Knowledge sharing among the team members facilitate an environment to improve team level output, especially it facilitates Agile software development dynamics through effective team interactions and thereby improves Tacit

knowledge capture and sharing. However, the way of eliciting complete Tacit knowledge from an expert is still an unanswered problem.

"The only irreplaceable capital an organization possesses is the knowledge and ability of its people. The productivity of that capital depends on how effectively people share their competence with those who can use it." - Andrew Carnegie, Scottish industrialist.

Reference

1. Ikujiro Nonaka and Hirotaka Takeuchi, The knowledge-Creating Company: How Japanese Companies Create the Dynamic of Innovation, Oxford University Press **(1995)**

2. http://en.wikipedia.org/wiki/Agile_software_development **(2014)**

3. Etienne Wenger, Richard McDermott and William M. Snyder, Cultivating Communities of Practice, Harvard Business Review Press **(2002)**

4. Mike Konrad and Sandy Shrum, CMMi® or Agile Why Not Embrace Both!, Technical Note, Software Engineering Process Management, SEI **(2008)**

5. "Tacit Knowledge" versus "Explicit Knowledge" Approaches to Knowledge Management Practice by Ron Sanchez Professor of Management, Copenhagen Business School, http://research.fraserhealth.ca/ **(2014)**

6. http://www.12manage.com/methods_nonaka_seci.html **(2014)**

7. Historical Roots of Agile Methods: Where Did "Agile Thinking" Come From? By Noura Abbas, Andrew M. Gravell, and Gray B. Wills School of Electronics and Computer Science, University of Southampton, UK, http://core.kmi.open.ac.uk/ **(2014)**

8. Mike Konrad and Sandy Shrum, CMMi® or Agile Why Not Embrace Both!, Technical Note, Software Engineering Process Management, SEI **(2008)**

9. http://agilemanifesto.org **(2014)**

10. Peter Anthony Busch, Knowledge Management Implications of Articulable Tacit Knowledge: Case Studies on its Diffusion – Thesis **(2004)**

11. http://articles.economictimes.indiatimes.com/2013-06-07/news/39815456_1_three-employees-indian-employees-attrition **(2014)**